Red Hat AMQ 2021.Q1

OpenShift への AMQ Interconnect のデプロイ

AMQ Interconnect 1.10 向け

Last Updated: 2023-02-27

Red Hat AMQ 2021.Q1 OpenShift への AMQ Interconnect のデプロイ

AMQ Interconnect 1.10 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 Interconnect 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 OPENSHIFT CONTAINER PLATFORM での AMQ INTERCONNECT の使用

AMQ Interconnect は、ハイブリッドクラウドおよび IoT/エッジデプロイメント向けに、大規模で回復性の高いメッセージングネットワークを構築するための軽量な AMQP 1.0 メッセージルーターです。AMQ Interconnect は、メッセージングエンドポイント (クライアント、サーバー、メッセージブローカーなど) のアドレスを自動的に把握し、エンドポイント間でメッセージを柔軟にルーティングします。

本書では、AMQ Interconnect Operator および提供される **Interconnect** カスタムリソース定義 (CRD) を使用して、OpenShift Container Platform に AMQ Interconnect をデプロイする方法を説明します。CRD は AMQ Interconnect デプロイメントを定義し、Operator は OpenShift Container Platform でデプロイメントを作成し、管理します。

1.1. OPERATOR とは

Operator は、Kubernetes アプリケーションのパッケージ化、デプロイメント、および管理を行う方法です。Operator は人間の運用上のナレッジを使用し、これをコンシューマーと簡単に共有できるソフトウェアにエンコードして、一般的なタスクや複雑なタスクを自動化します。

OpenShift Container Platform 4.0 では、**Operator Lifecycle Manager (OLM)** は、ユーザーがクラスター全体で実行されるすべての Operator やそれらの関連サービスをインストール、更新、およびそのライフサイクルを全般的に管理するのに役立ちます。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 Container Platform クラスター管理者が Operator を検出し、インストールし、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イスです。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。

関連情報

- Operator についての詳細は、OpenShift のドキュメント を参照してください。

1.2. 提供されるカスタムリソース

AMQ Interconnect Operator は **Interconnect** カスタムリソース定義 (CRD) を提供します。これにより、他の OpenShift Container Platform API オブジェクトと同様に、OpenShift Container Platform で実行されている AMQ Interconnect デプロイメントと対話できます。

Interconnect CRD は AMQ Interconnect ルーターデプロイメントを表します。CRD は、以下のように OpenShift Container Platform のルーターデプロイメントのさまざまな側面を定義する要素を提供します。

- AMQ Interconnect ルーターの数

- デプロイメントトポロジー
- 接続性
- アドレスセマンティクス

第2章 OPENSHIFT CONTAINER PLATFORM に AMQ INTERCONNECT をデプロイするための準備

AMQ Interconnect を OpenShift Container Platform にデプロイする前に、以下のいずれかの手順を実行します。

- 「Red Hat Integration - AMQ Certificate Manager Operator の追加」
- 「SSL/TLS 認証用のシークレットの作成」

AMQ Interconnect を評価する場合、この手順をスキップすることができますが、Red Hat は必ず AMQ Interconnect の通信をセキュアにすることを推奨します。

2.1. SSL/TLS 認証用のシークレットの作成

注記

Red Hat Integration - AMQ Certificate Manager Operator をインストールした場合、この手順は省略できます。AMQ Certificate Manager でネットワークのセキュリティーを保護する手順は、関連する手順に含まれます。OpenShift は **Secrets** というオブジェクトを使用して、SSL/TLS 証明書などの機密情報を保持します。ルーター間のトラフィック、クライアントトラフィック、またはその両方のセキュリティーを保護する場合、SSL/TLS 証明書と秘密鍵を作成し、それらをシークレットとして OpenShift に提供する必要があります。

手順

1. ルーター間接続用に既存の認証局 (CA) 証明書がない場合は、これを作成します。
以下のコマンドは、ルーター間接続用の自己署名 CA 証明書を作成します。

```
# Create a new directory for the inter-router certificates.
$ mkdir internal-certs

# Create a private key for the CA.
$ openssl genrsa -out internal-certs/ca-key.pem 2048

# Create a certificate signing request for the CA.
$ openssl req -new -batch -key internal-certs/ca-key.pem -out internal-certs/ca-csr.pem

# Self sign the CA certificate.
$ openssl x509 -req -in internal-certs/ca-csr.pem -signkey internal-certs/ca-key.pem -out
internal-certs/ca.crt
```

2. CA が署名したルーターの証明書を作成します。
これらのコマンドは、秘密鍵と証明書を作成し、直前の手順で作成した CA を使用して証明書に署名します。

```
# Create a private key.
$ openssl genrsa -out internal-certs/tls.key 2048

# Create a certificate signing request for the router.
$ openssl req -new -batch -subj "/CN=amq-interconnect.<project_name>.svc.cluster.local" -
key internal-certs/tls.key -out internal-certs/server-csr.pem
```

```
# Sign the certificate using the CA.
$ openssl x509 -req -in internal-certs/server-csr.pem -CA internal-certs/ca.crt -CAkey
internal-certs/ca-key.pem -out internal-certs/tls.crt -CAcreateserial
```

ここで、**<project_name>** は現在の OpenShift プロジェクトの名前です。

3. 秘密鍵、ルーター証明書、および CA 証明書を含むシークレットを作成します。
このコマンドは、直前の手順で作成した鍵および証明書を使用してシークレットを作成します。

```
$ oc create secret generic inter-router-certs-secret --from-file=tls.crt=internal-certs/tls.crt  --
from-file=tls.key=internal-certs/tls.key  --from-file=ca.crt=internal-certs/ca.crt
```

4. (SASL でクライアントを認証するのではなく) クライアント接続の認証に SSL/TLS を使用する場合は、クライアント接続用に CA 証明書を作成します。
以下のコマンドは、クライアント接続用の自己署名 CA 証明書を作成します。

```
# Create a new directory for the client certificates.
$ mkdir client-certs

# Create a private key for the CA.
$ openssl genrsa -out client-certs/ca-key.pem 2048

# Create a certificate signing request for the CA.
$ openssl req -new -batch -key client-certs/ca-key.pem -out client-certs/ca-csr.pem

# Self sign the certificate.
$ openssl x509 -req -in client-certs/ca-csr.pem -signkey client-certs/ca-key.pem -out client-
certs/ca.crt
```

5. CA によって署名されたクライアント接続用の証明書を作成します。
これらのコマンドは、秘密鍵と証明書を作成し、続いて直前の手順で作成した CA を使用して証明書に署名します。

```
# Create a private key.
$ openssl genrsa -out client-certs/tls.key 2048

# Create a certificate signing request for the client connections
$ openssl req -new -batch -subj "/CN=<client_name>" -key client-certs/tls.key -out client-
certs/client-csr.pem

# Sign the certificate using the CA.
$ openssl x509 -req -in client-certs/client-csr.pem -CA client-certs/ca.crt -CAkey client-
certs/ca-key.pem -out client-certs/tls.crt -CAcreateserial
```

ここで、**<client_name>** はそれぞれのルータークライアントについて一意です。

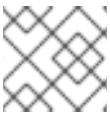
6. 直前の手順で作成した証明書を使用してクライアント証明書に署名するために使用される CA 証明書を含むシークレットを作成します。

```
$ oc create secret generic client-ca-secret --from-file=ca.crt=client-certs/ca.crt --from-
file=tls.crt=client-certs/ca.crt --from-file=tls.key=client-certs/ca-key.pem
```

2.2. RED HAT INTEGRATION - AMQ CERTIFICATE MANAGER OPERATOR の追加

Red Hat Integration - AMQ Certificate Manager Operator (cert-manager) は、TLS 証明書を発行および管理するためのオプションの Kubernetes アドオンです。Red Hat Integration - AMQ Interconnect Operator は、それを使用してルーターネットワークのセキュリティー保護に必要な TLS 証明書を自動的に作成します。

OperatorHub を使用して Operator を OpenShift Container Platform クラスターに追加します。



注記

Operator をインストールするには、OpenShift クラスターの管理者権限が必要です。

Red Hat Integration - AMQ Certificate Manager Operator は、OpenShift Container Platform 4.9 以降ではサポートされていません。または、「SSL/TLS 認証用のシークレットの作成」 の説明に従って TLS 証明書を作成し、管理できます。

インストールすると、Operator はクラスター内のすべてのユーザーおよびプロジェクトで利用可能になります。

前提条件

- **cluster-admin** アカウントにより OpenShift Container Platform 4.6、4.7、4.8、4.9 または 4.10 クラスターにアクセスできること。

手順

1. OpenShift Container Platform の Web コンソールで、**Operators → OperatorHub** に移動します。
2. 利用可能な Operator の一覧から **Red Hat Integration - AMQ Certificate Manager Operator** を選択し、**Install** をクリックします。
3. **Operator Installation** ページで、**All namespaces on the cluster (default)**を選択してから **Install** をクリックします。
   Installed Operators ページが表示され、Operator インストールのステータスが表示されます。
4. Verify that the Red Hat Integration - AMQ Certificate Manager Operator が表示されることを確認し、**Status** が **Succeeded** に変わるまで待機します。
5. インストールに成功しない場合は、エラーのトラブルシューティングを行います。
   a. **Installed Operators** ページで **Red Hat Integration - AMQ Certificate Manager Operator** をクリックします。
   b. **Subscription** タブを選択し、エラーを表示します。

関連情報

- **cert-manager** の詳細は、cert-manager のドキュメント を参照してください。

第3章 RED HAT INTEGRATION - AMQ INTERCONNECT OPERATOR の追加

Red Hat Integration - AMQ Interconnect Operator は、OpenShift Container Platform に AMQ Interconnect ルーターネットワークを作成し、管理します。この Operator は、これを使用するプロジェクトごとに個別にインストールする必要があります。

Operator をインストールするためのオプションは次のとおりです。

- 「CLI を使用した Operator のインストール」
- 「Operator Lifecycle Manager を使用した Operator のインストール」

注記

Operator をインストールするには、OpenShift クラスターの管理者権限が必要です。

3.1. CLI を使用した OPERATOR のインストール

本セクションの手順では、OpenShift コマンドラインインターフェイス (CLI) を使用して、指定の OpenShift プロジェクトに Red Hat Integration - AMQ Interconnect Operator の最新バージョンをインストールし、デプロイする方法を説明します。

3.1.1. Operator コードの取得

この手順では、最新バージョンの AMQ Interconnect 1.10 用 Operator をインストールするのに必要なコードにアクセスし、準備する方法を説明します。

手順

1. Web ブラウザーで、AMQ Interconnect リリース の **Software Downloads** ページに移動します。
2. **Version** ドロップダウンリストの値が **1.10.7** に設定され、**Releases** タブが選択されていることを確認します。
3. **AMQ Interconnect 1.10.7 Operator Installation and Example Files**の横にある **Download** をクリックします。
   amq-interconnect-operator-1.10.7-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します。
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 ~/**router**/**operator** という名前のディレクトリーに移動します。

```
$ mkdir ~/router
$ mv amq-interconnect-operator-1.10.7-ocp-install-examples.zip ~/router
```

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 ~/router
$ unzip amq-interconnect-operator-1.10.7-ocp-install-examples.zip
```

6. アーカイブの展開時に作成されたディレクトリーに移動します。以下に例を示します。

```
$ cd operator
```

7. 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
```

8. 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>
```

9. Operator で使用するサービスアカウントを作成します。

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

10. Operator のロールを作成します。

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

11. Operator のロールバインディングを作成します。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。

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

以下の手順では、Operator をプロジェクトにデプロイします。

3.1.2. CLI を使用した Operator のデプロイ

本セクションの手順では、OpenShift コマンドラインインターフェイス (CLI) を使用して、OpenShift プロジェクトに最新バージョンの AMQ Interconnect 1.10 用 Operator デプロイする方法を説明します。

前提条件

- Operator デプロイメントのために OpenShift プロジェクトを準備している必要がある。「Operator コードの取得」を参照してください。
- 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ります。

手順

1. OpenShift コマンドラインインターフェイス (CLI) で、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
```

2. Operator デプロイメント用に以前に準備したプロジェクトに切り替えます。以下に例を示します。

```
$ oc project <project-name>
```

3. 以前の手順で Operator インストールアーカイブを展開する際に作成されたディレクトリーに移動します。以下に例を示します。

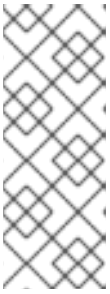
```
$ cd ~/router/operator/qdr-operator-1.10-ocp-install-examples
```

4. Operator に含まれる CRD をデプロイします。Operator をデプロイし、起動する前に、CRD を OpenShift クラスターにインストールする必要があります。

```
$ oc create -f deploy/crds/interconnectedcloud_v1alpha1_interconnect_crd.yaml
```

5. Red Hat Ecosystem Catalog での認証に使用され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 **default**、**deployer**、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 Interconnect 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。

6. Operator をデプロイします。

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

7. Operator が実行されていることを確認します。

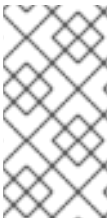
```
$ oc get pods -l name=qdr-operator
```

出力が Pod が実行されていることを報告しない場合は、以下のコマンドを使用して、実行を妨げている問題を判別します。

```
$ oc describe pod -l name=qdr-operator
```

8. CRD がクラスターに登録されていることを確認し、CRD の詳細を確認します。

```
$ oc get crd
$ oc describe crd interconnects.interconnectedcloud.github.io
```



注記

所定の OpenShift プロジェクトに AMQ Interconnect Operator の **単一インスタンス** のみをデプロイすることが推奨されます。Operator デプロイメントの **replicas** 要素を **1** より大きい値に設定したり、同じプロジェクトで Operator を 複数回デプロイしたりすることは **推奨されません**。

関連情報

- OperatorHub グラフィカルインターフェイスを使用する AMQ Interconnect Operator の別のインストール方法は、「Operator Lifecycle Manager を使用した Operator のインストール」を参照してください。

3.2. OPERATOR LIFECYCLE MANAGER を使用した OPERATOR のインストール

本セクションの手順では、OperatorHub を使用して、指定の OpenShift プロジェクトに Red Hat Integration - AMQ Interconnect Operator の最新バージョンをインストールし、デプロイする方法を説明します。

OpenShift Container Platform 4.1 移行では、**Operator Lifecycle Manager** (OLM) は、ユーザーがクラスター全体で実行されるすべての Operator やそれらの関連サービスをインストール、更新、およびそのライフサイクルを全般的に管理するのに役立ちます。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。

前提条件

- **cluster-admin** アカウントにより OpenShift Container Platform 4.6、4.7、4.8、4.9 または 4.10 クラスターにアクセスできること。
- 必要に応じて、Red Hat Integration - AMQ Certificate Manager Operator が OpenShift Container Platform クラスターにインストールされている。

手順

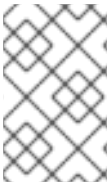
1. OpenShift Container Platform の Web コンソールで、**Operators → OperatorHub** に移動します。
2. 利用可能な Operator の一覧から **Red Hat Integration - AMQ Interconnect Operator** を選択し、**Install** をクリックします。
3. **Operator Installation** ページで、Operator をインストールする名前空間を選択し、続いて **Install** をクリックします。
   Installed Operators ページが表示され、Operator インストールのステータスが表示されます。
4. AMQ Interconnect Operator が表示されていることを確認し、**Status** が **Succeeded** に変更されるまで待機します。
5. インストールに成功しない場合は、エラーのトラブルシューティングを行います。
   a. **Installed Operators** ページで **Red Hat Integration - AMQ Interconnect Operator** をクリックします。
   b. **Subscription** タブを選択し、エラーを表示します。

第4章 制限された環境での RED HAT INTEGRATION - AMQ INTERCONNECT OPERATOR のインストール

インターネットアクセスがない、または制限されている実稼働環境で、3章*Red Hat Integration - AMQ Interconnect Operator の追加* の説明し従い Red Hat Integration - AMQ Interconnect Operator をインストールしすることはできません。このセクションでは、必要なコンポーネントをクラスターにミラーリングして、制限された環境で Red Hat Integration - AMQ Interconnect Operator をインストールする方法を説明します。

前提条件

- OpenShift Container Platform クラスターのバージョン 4.6、4.7、4.8、4.9、または 4.10
- 以下を備えた RHEL マシン:
  - **podman** バージョン 1.9.3 以降
  - OpenShift documentation で説明されているとおりの **opm** CLI
  - **oc** CLI バージョン 4.9.9 以降
- ネットワークアクセス
  - Red Hat Container レジストリーへのネットワークアクセス
  - OpenShift Container Platform クラスターへのネットワークアクセス



注記

ミラーリング時にのみ Red Hat Container レジストリーへのアクセスが必要です。Red Hat Container レジストリーと OpenShift Container Platform クラスターに同時にアクセスする必要はありません。

必要な手順は、以下のセクションで説明しています。

- 「OpenShift Container Platform クラスターのセットアップ」
- 「RHEL マシンでの AMQ Interconnect イメージの作成」
- 「OpenShift Container Platform クラスターへのイメージのプッシュ」

4.1. OPENSHIFT CONTAINER PLATFORM クラスターのセットアップ

OpenShift Container Platform クラスターで以下の手順を実行し、ミラーリングプロセスを準備します。

1. **cluster-admin** としてクラスターにログインします。
2. CLI または OpenShift コンソールを使用して、デフォルトカタログのソースを無効にします。
   a. CLI の場合、OperatorHub に **disableAllDefaultSources: true** を設定します。

```
$ oc patch OperatorHub cluster --type json \
-p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

b. OpenShift コンソールの場合、**Administration → Cluster Settings → Configuration → OperatorHub** に移動します。
OperatorHub ページで、Sources タブをクリックし、ソースを無効にします。

4.2. RHEL マシンでの AMQ INTERCONNECT イメージの作成

RHEL マシンで次の手順を実行し、ミラーリングプロセスを準備します。

前提条件

- **registry.redhat.io** へのアクセス

1. RHEL マシンから **registry.redhat.io** にログインします。

```
$ podman login -u USERNAME -p PASSWORD registry.redhat.io
```

2. Operator のリストに Interconnect Operator のみを保持します。

```
$ opm index prune -f registry.redhat.io/redhat/redhat-operator-index:v<openshift-version>
-p amq7-interconnect-operator -t <cluster-domain>:<registry-port>/iib:my-operator-iib
```

ここでは、以下のようになります。

- <openshift-version> は OpenShift Container Platform のバージョン (例: **4.9**) です。
- <cluster-domain> は OpenShift Container Platform クラスターのドメイン名です (例: **mycluster.example.com)**。
- <registry-port> は OpenShift Container Platform クラスターのレジストリーで使用されるポート番号で、デフォルトは **5000** です。

Interconnect Operator の podman イメージのみ作成したことを確認します。

```
$ podman images | grep my-operator-iib <cluster-domain>:<registry-port>/iib

my-operator-iib  39b6148e6981  3 days ago     138 MB
```

4.3. OPENSHIFT CONTAINER PLATFORM クラスターへのイメージのプッシュ

前提条件

- RHEL マシンから OpenShift Container Platform クラスターへのアクセス。

1. RHEL マシンから、イメージをクラスターレジストリーにプッシュします。

```
$ podman push <cluster-domain>:<registry-port>/iib:my-operator-iib
```

2. ミラーリングプロセスに必要な 3 つのファイルを作成します。

```
$ /usr/local/bin/oc adm catalog mirror  \
  <cluster-domain>:<registry-port>/iib:my-operator-iib \
  <cluster-domain>:<registry-port> \
```

```
  -a /home/customer-user/.docker/config.json \
  --insecure=true \
  --registry-config /home/customer-user/.docker/config.json \
  --index-filter-by-os=linux/amd64 \
  --manifests-only
```

3. 次のファイルが存在することを確認します。
   - **catalogSource.yaml** - catalogSource を記述する YAML ファイル。
   - **imageContentSourcePolicy.yaml** - 内部レジストリーのイメージを RedHat レジストリーからのアドレスにマップする YAML ファイル。
   - **mapping.txt** - イメージの内部レジストリーへのミラーリングプロセスを駆動するテキストファイル。
4. **mapping.txt** を編集して、ミラーリングするイメージのみを一覧表示します。
   ファイルの形式は次のとおりです。

   ```
   [ Operator address on RedHat registry : Operator SHA ] = [ Operator address on internal mirror registry : tag ]
   ```

   mapping.txt ファイルの例:

   ```
   registry.redhat.io/amq7/amq-interconnect@sha256:6101cc735e4d19cd67c6d80895c425ecf6f1d2604d88f999fa0cae57a7d6abaf=<cluster-domain>:<registry-port>/amq7-amq-interconnect:f793b0cc
   registry.redhat.io/amq7/amq-interconnect-operator@sha256:8dd53290c909589590b88a1544d854b4ad9f8b4a639189597c0a59579bc60c40=<cluster-domain>:<registry-port>/amq7-amq-interconnect-operator:73c142ff
   registry.redhat.io/amq7/amq-interconnect-operator-metadata@sha256:799ce48905d5d2a91b42e2a7943ce9b756aa9da80f6924be06b2a6275ac90214=<cluster-domain>:<registry-port>/amq7-amq-interconnect-operator-metadata:14cc4a4e
   ```

5. 必須イメージのミラーリング

   ```
   $ /usr/local/bin/oc image mirror \
     -f mapping-ic.yaml \
     -a /home/customer-user/.docker/config.json \
     --insecure=true \
     --registry-config /home/customer-user/.docker/config.json \
     --keep-manifest-list=true
   ```

6. **ImageContentSourcePolicy** (ICSP) 名を設定します。
   imageContentSourcePolicy.yaml ファイルの name フィールドを設定します (例: **my-operator-icsp**)。

   ICSP スニペットの例:

   ```
   ---
   apiVersion: operator.openshift.io/v1alpha1
   kind: ImageContentSourcePolicy
   metadata:
     labels:
   ```

```
      operators.openshift.org/catalog: "true"
  name: my-operator-icsp
spec:
  repositoryDigestMirrors:
  - mirrors:
    - <cluster-domain>:<registry-port>/amq7-amq-interconnect-operator
    source: registry.redhat.io/amq7/amq-interconnect-operator
```

7. ポリシー (ICSP) ファイルを適用します。

```
$ /usr/local/bin/oc create -f imageContentSourcePolicy.yaml
```

このファイルを適用すると、すべてのクラスターノードが自動的にリセットされます。**oc get nodes** を使用するか、OpenShift コンソールで **Compute → Nodes** に移動して、ノードのステータスを確認できます。

注記

続行する前に、すべてのノードが **Ready** 状態であることを確認してください。

8. catalogSource 名を設定します。
catalogSource.yaml ファイルの **name** フィールドを設定します (例: **my-operator-catalog** の)。

catalogSource.yaml ファイルの例:

```
apiVersion: operators.coreos.com/v1alpha1
kind: CatalogSource
metadata:
  name: iib
  namespace: openshift-marketplace
spec:
  image: <cluster-domain>:<registry-port>/iib:my-operator-iib
  sourceType: grpc
```

9. カタログソース設定を適用して、Red Hat Integration - AMQ Interconnect Operator のインストールを完了します。

```
$ /usr/local/bin/oc apply -f catalogSource.yaml
```

10. 「内部ルーターデプロイメントの作成」 の説明に従ってルーターを展開して、インストールが機能していることを確認します。

第5章 ルーターネットワークの作成

AMQ Interconnect ルーターのネットワークを作成するには、**Interconnect** カスタムリソースでデプロイメントを定義し、これを適用します。AMQ Interconnect Operator は、必要な Pod をスケジュールし必要なリソースを作成してデプロイメントを作成します。

本セクションの手順は、以下のルーターネットワークトポロジーを示しています。

- 内部ルーターメッシュ
- スケーラビリティー向けのエッジルーターを持つ内部ルーターメッシュ
- 2 つの OpenShift クラスターに接続するクラスター間ルーターネットワーク

前提条件

- AMQ Interconnect Operator が OpenShift Container Platform プロジェクトにインストールされている。

5.1. 内部ルーターデプロイメントの作成

内部ルーターは相互に接続を確立し、ネットワーク全体で最小コストのパスを自動的に算出します。

手順

以下の手順では、3 つのルーターの内部ルーターネットワークを作成します。ルーターはメッシュトポロジーで自動的に相互に接続し、それらの接続は相互 SSL/TLS 認証で保護されます。

1. 内部ルーターデプロイメントを記述する **Interconnect** カスタムリソース YAML ファイルを作成します。

   router-mesh.yaml ファイルのサンプル

```
apiVersion: interconnectedcloud.github.io/v1alpha1
kind: Interconnect
metadata:
  name: router-mesh
spec:
  deploymentPlan:
    role: interior (1)
    size: 3 (2)
    placement:  Any (3)
```

   (1) デプロイメント内のルーターの操作モード。Operator はメッシュトポロジーの内部ルーターを自動的に接続します。

   (2) 作成するルーターの数。

   (3) 各ルーターは別個の Pod で実行されます。配置は、Operator が Pod をスケジュールして配置するクラスターの場所を定義します。以下の配置オプションを選択できます。

   Any
   Pod は OpenShift Container Platform クラスターの任意のノードで実行できます。

   Every

Operator はルーター Pod をクラスター内の各ノードに配置します。このオプションを選択すると、**Size** プロパティーは必要ありません。ルーター数はクラスター内のノード数に対応します。

AntiAffinity

Operator は、複数のルーター Pod がクラスター内の同じノードで実行されないようにします。サイズがクラスター内のノードの数を超える場合、スケジュールできない追加の Pod は **Pending** 状態のままになります。

2. YAML ファイルに記述されるルーターデプロイメントを作成します。

```
$ oc apply -f router-mesh.yaml
```

Operator は、デフォルトのアドレスセマンティクスを使用するメッシュトポロジーの内部ルーターデプロイメントを作成します。また、ルーターにアクセスできるサービスと、Web コンソールにアクセスできるルートも作成します。

3. ルーターメッシュが作成され、Pod が実行されていることを確認します。
各ルーターは別個の Pod で実行されます。これらは、Operator が作成したサービスを使用して相互に自動接続します。

```
$ oc get pods
NAME                                    READY   STATUS    RESTARTS   AGE
interconnect-operator-587f94784b-4bzdx  1/1     Running   0          52m
router-mesh-6b48f89bd-588r5             1/1     Running   0          40m
router-mesh-6b48f89bd-bdjc4             1/1     Running   0          40m
router-mesh-6b48f89bd-h6d5r             1/1     Running   0          40m
```

4. ルーターデプロイメントを確認します。

```
$ oc get interconnect/router-mesh -o yaml
apiVersion: interconnectedcloud.github.io/v1alpha1
kind: Interconnect
...
spec:
  addresses: 1
  - distribution: closest
    prefix: closest
  - distribution: multicast
    prefix: multicast
  - distribution: closest
    prefix: unicast
  - distribution: closest
    prefix: exclusive
  - distribution: multicast
    prefix: broadcast
  deploymentPlan: 2
    livenessPort: 8888
    placement: Any
    resources: {}
    role: interior
    size: 3
  edgeListeners: 3
  - port: 45672
  interRouterListeners: 4
```

```
- authenticatePeer: true
  expose: true
  port: 55671
  saslMechanisms: EXTERNAL
  sslProfile: inter-router
listeners:  5
- port: 5672
- authenticatePeer: true
  expose: true
  http: true
  port: 8080
- port: 5671
  sslProfile: default
sslProfiles:  6
- credentials: router-mesh-default-tls
  name: default
- caCert: router-mesh-inter-router-tls
  credentials: router-mesh-inter-router-tls
  mutualAuth: true
  name: inter-router
users: router-mesh-users  7
```

1 デフォルトのアドレス設定。これらの接頭辞のいずれにも一致しないアドレスに送信されたすべてのメッセージは、バランス型の anycast パターン で配布されます。

2 3 つの内部ルーターのルーターメッシュがデプロイされました。

3 それぞれの内部ルーターは、エッジルーターからの接続をポート **45672** でリッスンします。

4 内部ルーターは、ポート **55671** で相互に接続します。これらのルーター間の接続は、SSL/TLS 相互認証で保護されます。**inter-router** SSL プロファイルには、Operator が生成した証明書の詳細が含まれます。

5 それぞれの内部ルーターは、以下のポートで外部クライアントからの接続をリッスンします。

- **5672**: メッセージングアプリケーションからの非セキュアな接続
- **5671**: メッセージングアプリケーションからのセキュアな接続
- **8080**: AMQ Interconnect Web コンソールへのアクセス。デフォルトのユーザー名/パスワードのセキュリティーが適用されます。

6 Red Hat Integration - AMQ Certificate Manager Operator を使用すると、Red Hat Integration - AMQ Interconnect Operator は 2 つの SSL プロファイルを自動的に作成します。

- **inter-router**: Operator は、認証局 (CA) を作成し、各内部ルーターの CA によって署名された証明書を生成し、相互 TLS 認証を使用してルーター間のネットワークを保護します。
- **default**: Operator は、ポート **5671** で内部ルーターに接続するために、メッセージングアプリケーションの TLS 証明書を作成します。

7 AMQ Interconnect Web コンソールは、ユーザー名/パスワードの認証でセキュア化されます。Operator は認証情報を自動的に生成し、それらを **router-mesh-users** Secret に保存

します。

5.2. エッジルーターデプロイメントの作成

エッジルーターデプロイメントを追加して、ルーターネットワークを効率的にスケーリングできます。エッジルーターは、メッセージングアプリケーションの接続コンセントレーターとして機能します。各エッジルーターは、内部ルーターへの単一のアップリンク接続を維持し、メッセージングアプリケーションはエッジルーターに接続してメッセージを送受信します。

前提条件

- 内部ルーターメッシュがデプロイされている。詳細は、「内部ルーターデプロイメントの作成」を参照してください。

手順

この手順では、OpenShift Container Platform クラスターの各ノードにエッジルーターを作成し、それらを以前に作成した内部ルーターメッシュに接続します。

1. エッジルーターデプロイメントを記述する **Interconnect** カスタムリソース YAML ファイルを作成します。

   edge-routers.yaml ファイルのサンプル

```
apiVersion: interconnectedcloud.github.io/v1alpha1
kind: Interconnect
metadata:
  name: edge-routers
spec:
  deploymentPlan:
    role: edge
    placement: Every (1)
  edgeConnectors: (2)
    - host: router-mesh (3)
      port: 45672 (4)
```

   (1) エッジルーター Pod は、OpenShift Container Platform クラスターの各ノードにデプロイされます。この配置は、クラスター全体でメッセージングアプリケーショントラフィックのバランスを取るのに役立ちます。Operator は DaemonSet を作成し、スケジュールされる Pod の数がクラスター内のノード数を常に表すようにします。

   (2) エッジコネクターは、エッジルーターから内部ルーターへの接続を定義します。

   (3) 内部ルーター用に作成されたサービスの名前。

   (4) 内部ルーターがエッジ接続をリッスンするポート。デフォルトは **45672** です。

2. YAML ファイルで記述されるエッジルーターを作成します。

```
$ oc apply -f edge-routers.yaml
```

   Operator は OpenShift Container Platform クラスターの各ノードにエッジルーターをデプロイし、それらを内部ルーターに接続します。

3. エッジルーターが作成され、Pod が実行されていることを確認します。
   各ルーターは別個の Pod で実行されます。各エッジルーターは、以前に作成された内部ルーターのいずれかに接続します。

```
$ oc get pods
NAME                                    READY   STATUS    RESTARTS   AGE
edge-routers-2jz5j                      1/1     Running   0          33s
edge-routers-fhlxv                      1/1     Running   0          33s
edge-routers-gg2qb                      1/1     Running   0          33s
edge-routers-hj72t                      1/1     Running   0          33s
interconnect-operator-587f94784b-4bzdx  1/1     Running   0          54m
router-mesh-6b48f89bd-588r5             1/1     Running   0          42m
router-mesh-6b48f89bd-bdjc4             1/1     Running   0          42m
router-mesh-6b48f89bd-h6d5r             1/1     Running   0          42m
```

5.3. クラスター間ルーターネットワークの作成

AMQ Certificate Manager を使用しているかどうかに応じて、クラスター間ルーターネットワークの作成の手順が異なります。

- 「AMQ Certificate Manager を使用したクラスター間ルーターネットワークの作成」
- 「認証局を使用したクラスター間ルーターネットワークの作成」

5.3.1. 認証局を使用したクラスター間ルーターネットワークの作成

異なる OpenShift Container Platform クラスターで実行されているルーターからルーターネットワークを作成できます。これにより、別のクラスターで実行しているアプリケーションを接続できます。

前提条件

- 各ルーターの既存の証明書を定義するシークレットをすでに作成済みである。

手順

以下の手順では、2 つの異なる OpenShift Container Platform クラスター (**cluster1** および **cluster2**) にルーターデプロイメントを作成し、それらを接続してクラスター間のルーターネットワークを形成します。ルーターデプロイメント間の接続は、SSL/TLS 相互認証を使用して保護されます。

1. 最初の OpenShift Container Platform クラスター (**cluster1)** で、内部ルーターデプロイメントを記述する **Interconnect** カスタムリソース YAML ファイルを作成します。
   この例では、デフォルト設定で単一の内部ルーターを作成します。

   cluster1-router-mesh.yaml ファイルのサンプル

```
apiVersion: interconnectedcloud.github.io/v1alpha1
kind: Interconnect
metadata:
  name: cluster1-router-mesh
spec:
  interRouterListeners:
    - authenticatePeer: true (1)
      host: 0.0.0.0 (2)
      port: 55672 (3)
```

```
    saslMechanisms: EXTERNAL 4
    sslProfile: inter-router-profile 5
    expose: true 6
  sslProfiles:
   - caCert: inter-router-certs-secret 7
     credentials: inter-router-certs-secret 8
     name: inter-router-profile 9
```

1. TLS 証明書を使用して認証するには、**authenticatePeer** を **true** に設定する必要があります。
2. リスナーホスト
3. リスナーポート
4. 認証する SASL メカニズム。TLS 証明書には EXTERNAL を使用します。
5. クライアントの認証に使用する ssl-profile 名。
6. ポートにクラスター外からアクセスできるようにルートを公開します。
7. クラスターシークレットまたは **ca.crt** 名が含まれる (認証情報で使用されるのと同じシークレットを使用する場合、そうでなければ tls.crt が含まれている必要があります) CA の名前
8. クラスターシークレットおよび **tls.crt** および **tls.key** ファイルが含まれる CA 証明書の名前
9. interRouterListener に使用する ssl-profile 名

2. YAML ファイルに記述されるルーターデプロイメントを作成します。

```
$ oc apply -f cluster1-router-mesh.yaml
```

Red Hat Integration - AMQ Interconnect Operator は、デフォルト設定の内部ルーターおよび他のルーターを認証するためのリスナーを作成します。

3. 2 つ目の OpenShift Container Platform クラスター (**cluster2**) にログインし、2 つ目のルーターデプロイメントを作成するプロジェクトに切り替えます。

4. **cluster2** で、ルーターデプロイメントを記述する **Interconnect** カスタムリソース YAML ファイルを作成します。

cluster2-router-mesh.yaml ファイルのサンプル

```
apiVersion: interconnectedcloud.github.io/v1alpha1
kind: Interconnect
metadata:
  name: cluster2-router-mesh
spec:
  sslProfiles:
  - name: inter-router-profile 1
    credentials: inter-router-certs-secret
    caCert: inter-router-certs-secret
```

```
interRouterConnectors:
- host: cluster1-router-mesh-port-55672-myproject.cluster1.openshift.com (2)
  port: 443
  verifyHostname: false
  sslProfile: inter-router-profile
  name: cluster1
```

(1) この SSL プロファイルは、**cluster1** のルーターデプロイメントへの接続に必要な証明書を定義します。

(2) **cluster1** のルーター間リスナーの Route の URL。

5. YAML ファイルに記述されるルーターデプロイメントを作成します。

```
$ oc apply -f cluster2-router-mesh.yaml
```

6. ルーターが接続されていることを確認します。
この例では、**cluster2** のルーターから **cluster1** のルーターへの接続を表示しています。

```
$ oc exec cluster2-fb6bc5797-crvb6 -it -- qdstat -c
Connections
  id    host                                                container                role         dir
security                        authentication  tenant

================================================================================
================================================================================
================================================
  1     cluster1-router-mesh-port-55672-myproject.cluster1.openshift.com:443  cluster1-router-
mesh-54cffd9967-9h4vq  inter-router  out  TLSv1/SSLv3(DHE-RSA-AES256-GCM-SHA384)
x.509
```

5.3.2. AMQ Certificate Manager を使用したクラスター間ルーターネットワークの作成

異なる OpenShift Container Platform クラスターで実行されているルーターからルーターネットワークを作成できます。これにより、別のクラスターで実行しているアプリケーションを接続できます。

手順

以下の手順では、2 つの異なる OpenShift Container Platform クラスター (**cluster1** および **cluster2**) にルーターデプロイメントを作成し、それらを接続してクラスター間のルーターネットワークを形成します。ルーターデプロイメント間の接続は、SSL/TLS 相互認証を使用して保護されます。

1. 最初の OpenShift Container Platform クラスター (**cluster1)** で、内部ルーターデプロイメントを記述する **Interconnect** カスタムリソース YAML ファイルを作成します。
この例では、デフォルト設定で単一の内部ルーターを作成します。

cluster1-router-mesh.yaml ファイルのサンプル

```
apiVersion: interconnectedcloud.github.io/v1alpha1
kind: Interconnect
metadata:
  name: cluster1-router-mesh
spec: {}
```

2. YAML ファイルに記述されるルーターデプロイメントを作成します。

```
$ oc apply -f cluster1-router-mesh.yaml
```

Red Hat Integration - AMQ Interconnect Operator は、デフォルト設定で内部ルーターを作成します。Red Hat Integration - AMQ Certificate Manager Operator を使用して認証局 (CA) を作成し、CA が署名した証明書を生成します。

3. 2 つ目の OpenShift Container Platform クラスター (**cluster2**) でルーターデプロイメントの追加の証明書を生成します。
cluster2 のルーターデプロイメントには、**cluster1** の CA が発行する証明書が必要です。

a. 証明書を要求する **Certificate** カスタムリソース YAML ファイルを作成します。

certificate-request.yaml ファイルのサンプル

```
apiVersion: certmanager.k8s.io/v1alpha1
kind: Certificate
metadata:
  name: cluster2-inter-router-tls
spec:
  commonName: cluster1-router-mesh-myproject.cluster2.openshift.com
  issuerRef:
    name: cluster1-router-mesh-inter-router-ca (1)
  secretName: cluster2-inter-router-tls-secret
---
```

(1) **cluster1** のルーター間 CA を作成した発行者の名前。デフォルトでは発行者の名前は **<application-name>-inter-router-ca** です。

b. YAML ファイルで記述される証明書を作成します。

```
$ oc apply -f certificate-request.yaml
```

c. 生成した証明書を抽出します。

```
$ mkdir /tmp/cluster2-inter-router-tls
$ oc extract secret/cluster2-inter-router-tls-secret --to=/tmp/cluster2-inter-router-tls
```

4. 2 つ目の OpenShift Container Platform クラスター (**cluster2**) にログインし、2 つ目のルーターデプロイメントを作成するプロジェクトに切り替えます。

5. **cluster2** で、生成した証明書が含まれるシークレットを作成します。

```
$ oc create secret generic cluster2-inter-router-tls-secret --from-file=/tmp/cluster2-inter-router-tls
```

6. **cluster2** で、ルーターデプロイメントを記述する **Interconnect** カスタムリソース YAML ファイルを作成します。

cluster2-router-mesh.yaml ファイルのサンプル

```
apiVersion: interconnectedcloud.github.io/v1alpha1
```

```
kind: Interconnect
metadata:
  name: cluster2-router-mesh
spec:
  sslProfiles:
  - name: inter-cluster-tls (1)
    credentials: cluster2-inter-router-tls-secret
    caCert: cluster2-inter-router-tls-secret
  interRouterConnectors:
  - host: cluster1-router-mesh-port-55671-myproject.cluster1.openshift.com (2)
    port: 443
    verifyHostname: false
    sslProfile: inter-cluster-tls
```

(1) この SSL プロファイルは、**cluster1** のルーターデプロイメントへの接続に必要な証明書を定義します。

(2) **cluster1** のルーター間リスナーの Route の URL。

7. YAML ファイルに記述されるルーターデプロイメントを作成します。

```
$ oc apply -f cluster2-router-mesh.yaml
```

8. ルーターが接続されていることを確認します。
この例では、**cluster2** のルーターから **cluster1** のルーターへの接続を表示しています。

```
$ oc exec cluster2-fb6bc5797-crvb6 -it -- qdstat -c
Connections
  id    host                                                container                role         dir
security                        authentication  tenant

================================================================================
================================================================================
================================================
  1     cluster1-router-mesh-port-55671-myproject.cluster1.openshift.com:443  cluster1-router-
mesh-54cffd9967-9h4vq  inter-router  out  TLSv1/SSLv3(DHE-RSA-AES256-GCM-SHA384)
x.509
```

第6章 クライアントのルーターネットワークへの接続

ルーターネットワークの作成後に、クライアント (メッセージングアプリケーション) を接続し、メッセージの送受信を開始できるようにします。

デフォルトでは、Red Hat Integration - AMQ Interconnect Operator はルーターデプロイメント用にサービスを作成し、クライアントアクセス用に以下のポートを設定します。

- **5672**: 認証を行わないプレーンの AMQP トラフィック用
- **5671**: TLS 認証でセキュリティーが確保される AMQP トラフィック用

クライアントをルーターネットワークに接続するには、以下を実行できます。

- すべてのクライアントが OpenShift クラスター外にある場合は、ポートを公開し、ルーターネットワークに接続できるようにします。
- ルーターネットワークに接続するようにクライアントを設定します。

6.1. OPENSHIFT CONTAINER PLATFORM 外のクライアントへのポートの公開

ポートを公開し、OpenShift Container Platform クラスター外のクライアントがルーターネットワークに接続できるようにします。

手順

1. ポートを公開するルーターデプロイメントを記述する **Interconnect** カスタムリソース YAML ファイルの編集を開始します。

```
$ oc edit -f router-mesh.yaml
```

2. **spec.listeners** セクションで、クラスター外のクライアントがアクセスできるようにする各ポートを公開します。
この例では、ポート **5671** が公開されます。これにより、クラスター外のクライアントが、ルーターネットワークとの間で認証され、接続できるようになります。

router-mesh.yaml ファイルのサンプル

```
apiVersion: interconnectedcloud.github.io/v1alpha1
kind: Interconnect
metadata:
  name: router-mesh
spec:
  ...
  listeners:
    - port: 5672
    - authenticatePeer: true
      expose: true
      http: true
      port: 8080
    - port: 5671
```

```
      sslProfile: default
      expose: true
  ...
```

Red Hat Integration - AMQ Interconnect Operator は、クラスター外からのクライアントがルーターネットワークへの接続に使用できるルートを作成します。

6.2. クライアント接続の認証

ルーターデプロイメントを作成する場合、Red Hat Integration - AMQ Interconnect Operator は Red Hat Integration - AMQ Certificate Manager Operator を使用してクライアント認証用のデフォルト SSL/TLS 証明書を作成し、SSL 暗号化用にポート **5671** を設定します。

6.3. ルーターネットワークに接続するクライアントの設定

ルーターネットワークと同じ OpenShift クラスター、別のクラスター、または OpenShift 外部で実行されているメッセージングクライアントを接続して、メッセージを交換できるようにすることが可能です。

前提条件

- クライアントが OpenShift Container Platform クラスター外にある場合は、接続ポートを公開する必要があります。詳細は、「OpenShift Container Platform 外のクライアントへのポートの公開」を参照してください。

手順

- クライアントをルーターネットワークに接続するには、以下の接続 URL 形式を使用します。

  ```
  <scheme>://[<username>@]<host>[:<port>]
  ```

  <scheme>

  以下のいずれかを使用します。

  - **amqp**: 同じ OpenShift クラスター内からの暗号化されていない TCP
  - **amqps**: SSL/TLS 認証を使用したセキュアな接続用
  - **amqpws**: OpenShift クラスター外からの暗号化されていない接続用の AMQP over WebSockets

  <username>

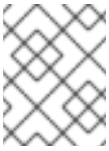
  ユーザー名/パスワード認証でルーターメッシュをデプロイした場合は、クライアントのユーザー名を指定します。

  <host>

  クライアントがルーターネットワークと同じ OpenShift クラスター内にある場合、OpenShift Service ホスト名を使用します。そうでない場合は、ルートのホスト名を使用します。

  <port>

  ルートに接続する場合は、ポートを指定する必要があります。セキュアでない接続に接続するには、ポート **80** を使用します。それ以外の場合は、セキュアな接続に接続するには、ポート **443** を使用します。



注記

セキュアでない接続 (ポート **80**) に接続するには、クライアントは AMQP over WebSockets (**amqpws**) を使用する必要があります。

以下の表は、接続 URL のサンプルを示しています。

URL	説明
amqp://admin@router-mesh:5672	クライアントとルーターネットワークはいずれも同じ OpenShift クラスターにあるので、接続 URL にサービスホスト名が使用されます。この場合、ユーザー名/パスワード認証が実装され、ユーザー名 (**admin**) を指定する必要があります。
amqps://router-mesh-myproject.mycluster.com:443	クライアントは OpenShift 外にあるため、接続 URL に Route ホスト名が使用されます。この場合、**amqps** スキームとポート **443** が必要な SSL/TLS 認証が実装されています。
amqpws://router-mesh-myproject.mycluster.com:80	クライアントは OpenShift 外にあるため、接続 URL に Route ホスト名が使用されます。この場合、認証は実装されないため、クライアントは **amqpws** スキームおよびポート **80** を使用する必要があります。

第7章 外部サービスへの接続

メッセージブローカーなどの外部サービスにルーターを接続できます。サービスはルーターネットワークと同じ OpenShift クラスターで実行されているか、または OpenShift 外で実行される可能性があります。

前提条件

- メッセージブローカーにアクセスできる必要があります。

手順

この手順では、ルーターをブローカーに接続し、メッセージングクライアントを接続するリンクルートを設定する方法を説明します。

1. ブローカーに接続するルーターデプロイメントを記述する **Interconnect** カスタムリソース YAML ファイルの編集を開始します。

```
$ oc edit -f router-mesh.yaml
```

2. **spec** セクションで、接続およびリンクルートを設定します。

router-mesh.yaml ファイルのサンプル

```
apiVersion: interconnectedcloud.github.io/v1alpha1
kind: Interconnect
metadata:
  name: router-mesh
spec:
  ...
  connectors: (1)
  - name: my-broker
    host: broker
    port: 5672
    routeContainer: true
  linkRoutes: (2)
  - prefix: q1
    direction: in
    connection: my-broker
  - prefix: q1
    direction: out
    connection: my-broker
```

(1) このルーターをメッセージブローカーに接続するために使用される接続。Operator は、このルーターデプロイメントで定義されたすべてのルーターからブローカーにこの接続を設定します。ルーターネットワークとブローカー間の単一の接続のみが必要な場合、コネクターの代わりに **リスナー** を設定し、ブローカーに接続を確立させます。

(2) リンクルート設定。これは、送受信リンクおよびメッセージングアプリケーションをメッセージブローカーに接続するために使用される接続を定義します。

3. ルーターがメッセージブローカーへのリンクルートを確立していることを確認します。

```
$ oc exec router-mesh-fb6bc5797-crvb6 -it -- qdstat --linkroutes
```

```
Link Routes
  address  dir  distrib       status
  ====================================
  q1       in   linkBalanced  active
  q1       out  linkBalanced  active
```

関連情報

- リンクルートの詳細は、Creating link routes を参照してください。

第8章 メッセージルーティング用のアドレス領域の設定

AMQ Interconnect は、柔軟なアプリケーション層のアドレス設定と配信セマンティクスを提供します。アドレスを設定することで、メッセージをエニーキャスト (最近接もしくはバランス型) またはマルチキャストパターンでルーティングできます。

8.1. クライアント間のメッセージのルーティング

デフォルトでは、AMQ Interconnect はメッセージをバランス型のエニーキャストパターンで配信します (それぞれのメッセージが単一のコンシューマーに配信され、AMQ Interconnect はトラフィックの負荷をネットワーク全体に分散しようとします)。そのため、デフォルト以外のセマンティクスをアドレスまたはアドレス範囲に適用する場合にしか、アドレス設定を変更する必要がありません。

手順

この手順では、マルチキャスト分散を使用するアドレスを設定します。ルーターネットワークは、このアドレスに送信された各メッセージのコピーを、アドレスにサブスクライブされているすべてのコンシューマーに配布します。

1. ルーターデプロイメントを記述する **Interconnect** カスタムリソース YAML ファイルの編集を開始します。

```
$ oc edit -f router-mesh.yaml
```

2. **spec** セクションで、アドレスに適用するセマンティクスを定義します。

router-mesh.yaml ファイルのサンプル

```
apiVersion: interconnectedcloud.github.io/v1alpha1
kind: Interconnect
metadata:
  name: router-mesh
spec:
  ...
  addresses:
  - pattern: */orders  (1)
    distribution: multicast
```

(1) **orders** で終了するアドレスに送信されたメッセージは、マルチキャストパターンで配布されます。

Operator は変更をルーターネットワークに適用し、各 Pod を再起動します。

3. ルーターネットワークのルーターを定義する追加のルーターデプロイメントのカスタムリソースがある場合、各 CR についてこの手順を繰り返します。
ルーターネットワーク内の各ルーターは、同じアドレス設定を持つ必要があります。

関連情報

- 設定可能なアドレスセマンティクスについての詳細は、Configuring message routing を参照してください。

8.2. ブローカーによるメッセージのルーティング

メッセージを保存および転送する必要がある場合は、メッセージブローカーのキューを介してそれらをルーティングすることができます。このシナリオでは、メッセージプロデューサーがメッセージをルーターに送信し、ルーターはメッセージをブローカーキューに送信します。コンシューマーがメッセージを受信するためにルーターに接続すると、ルーターはブローカーキューからメッセージを取得します。

ルーターネットワークと同じ OpenShift クラスターで実行されているブローカーにメッセージをルーティングすることも、クラスター外で実行されているブローカーにメッセージをルーティングすることもできます。

前提条件

- メッセージブローカーにアクセスできる必要があります。

手順

1. ルーターデプロイメントを記述する Interconnect カスタムリソース YAML ファイルの編集を開始します。

```
$ oc edit -f router-mesh.yaml
```

2. **spec** セクションに、ブローカーに接続するコネクター、ブローカーキューをポイントするウェイポイントアドレス、およびキューへのリンクを作成する自動リンクを追加します。

router-mesh.yaml ファイルのサンプル

```
apiVersion: interconnectedcloud.github.io/v1alpha1
kind: Interconnect
metadata:
  name: router-mesh
spec:
  ...
  addresses:
  - prefix: my-queue (1)
    waypoint: true
  autoLinks: (2)
  - address: my-queue
    direction: in
    connection: my-broker
  - address: my-queue
    direction: out
    connection: my-broker
  connectors: (3)
  - name: my-broker
    host: broker
    port: 5672
    routeContainer: true
```

(1) メッセージをブローカーキューに保存するアドレス (またはアドレスセット)。

(2) 自動リンク設定。これは、送受信リンクおよびブローカーのメッセージを送受信するために使用される接続を定義します。

(3) ルーターをメッセージブローカーに接続するために使用される接続

Operator は変更をルーターネットワークに適用し、各 Pod を再起動します。

3. ルーターがメッセージブローカーへの自動リンクを確立していることを確認します。

```
$ oc exec router-mesh-6d6dccb57f-x5cqf -it -- qdstat --autolinks
AutoLinks
  addr      dir  phs  extAddr  link  status  lastErr
  ================================================
  my-queue  in   1           26    active
  my-queue  out  0           27    active
```

4. ルーターネットワークのルーターを定義する追加のルーターデプロイメントのカスタムリソースがある場合、各 CR についてこの手順を繰り返します。
ルーターネットワーク内の各ルーターは、同じアドレス設定を持つ必要があります。

関連情報

- メッセージのブローカーキューへの、およびブローカーキューからのルーティングに関する詳細は、Routing Messages through broker queues を参照してください。

第9章 PROMETHEUS および GRAFANA を使用したルーターネットワークの監視

Prometheus は、履歴データを保存し、AMQ Interconnect などの大規模でスケーラブルなシステムを監視するために構築されたコンテナーネイティブなソフトウェアです。現在実行中のセッションだけでなく、長期間にわたるデータを収集します。

Prometheus および Alertmanager を使用して AMQ Interconnect データを監視および保存することで、Grafana などのグラフィカルツールを使用してデータを視覚化し、クエリーを実行することができます。

9.1. PROMETHEUS および GRAFANA の設定

AMQ Interconnect ダッシュボードを表示する前に、AMQ Interconnect がデプロイされた OpenShift プロジェクトに Prometheus、Alertmanager、および Grafana をデプロイおよび設定する必要があります。必要な設定ファイルはすべて GitHub リポジトリーで提供されます。

手順

1. **qdr-monitoring** GitHub リポジトリー のクローンを作成します。
   このリポジトリーには、AMQ Interconnect を監視するために Prometheus および Grafana を設定するために必要な設定ファイルのサンプルが含まれています。

   ```
   $ git clone https://github.com/interconnectedcloud/qdr-monitoring
   ```

2. **NAMESPACE** 環境変数を AMQ Interconnect をデプロイしたプロジェクトの名前に設定します。
   たとえば、**example** プロジェクトに AMQ Interconnect をデプロイした場合は、以下のように **NAMESPACE** 環境変数を設定します。

   ```
   $ export NAMESPACE=example
   ```

3. **deploy-monitoring.sh** スクリプトを実行します。
   このスクリプトは、Prometheus、Alertmanager、および Grafana を OpenShift プロジェクトにデプロイするために必要な OpenShift リソースを作成および設定します。また、ルーターネットワークのメトリクスを提供する 2 つのダッシュボードも設定します。

   ```
   $ ./deploy-monitoring.sh
   ```

   このスクリプトを実行する代替方法として、ターゲットプロジェクトをパラメーターとして指定します。以下に例を示します。

   ```
   $ ./deploy-monitoring.sh example
   ```

関連情報

- Prometheus の詳細は、Prometheus のドキュメント を参照してください。
- Grafana の詳細は、Grafana のドキュメント を参照してください。

9.2. GRAFANA での AMQ INTERCONNECT ダッシュボードの表示

Prometheus および Grafana の設定後に、以下の Grafana ダッシュボードで AMQ Interconnect データを可視化できます。

Qpid Dispatch Router
: 以下のメトリクスを表示します。

Qpid Dispatch Router
: 以下のメトリクスを表示します。

- **Deliveries ingress**
- **Deliveries egress**
- **Deliveries ingress route container**
- **Deliveries egress route container**
- **Deliveries redirected to fallback destination**
- **Dropped presettled deliveries**
- **Presettled deliveries**
- **Auto links**
- **Link routes**
- **Address count**
- **Connection count**
- **Link count**

Qpid Dispatch Router - Delayed Deliveries
: 以下のメトリクスを表示します。

- **Cumulative delayed 10 seconds**
- **Cumulative delayed 1 second**
- **Rate of new delayed deliveries**

これらのメトリクスについての詳細は、「ルーターメトリクス」を参照してください。

手順

1. OpenShift Web コンソールで、**Networking → Routes** に切り替えて、**grafana** ルート の URL をクリックします。
   Grafana のログインページが表示されます。

2. ユーザー名とパスワードを入力し、続いて **Log In** をクリックします。
   デフォルトの Grafana ユーザー名およびパスワードは、どちらも **admin** です。初回ログイン後に、パスワードを変更できます。

3. 上部のヘッダーでダッシュボードのドロップダウンメニューをクリックし、**Qpid Dispatch Router** または **Qpid Dispatch Router - Delayed Deliveries** ダッシュボードを選択します。

図9.1 Delayed Deliveries ダッシュボード

9.3. ルーターメトリクス

以下のメトリクスは Prometheus で利用できます。

qdr_connections_total

ルーターへのネットワーク接続の総数。これには、AMQP ルートコンテナーから、および AMQP ルートコンテナーへの接続が含まれます。

qdr_links_total

ルーターにアタッチされた受信および発信リンクの総数。

qdr_addresses_total

ルーターが認識するアドレスの総数。

qdr_routers_total

ルーターが認識するルーターの総数。

qdr_link_routes_total

ルーターに設定されたアクティブおよび非アクティブなリンクルートの総数。詳細は、Understanding link routing を参照してください。

qdr_auto_links_total

ルーターに設定された受信リンクおよび発信自動リンクの総数。自動リンクの詳細は、Configuring brokered messaging を参照してください。

qdr_presettled_deliveries_total

ルーターに到達する事前処理済み配信の総数。ルーターは受信配信を処理し、処理をメッセージの送付先に反映させます (**fire and forget** とも呼ばれます)。

qdr_dropped_presettled_deliveries_total

ルーターが輻輳のために破棄する事前処理済みの配信の総数。ルーターは受信配信を処理し、処理をメッセージの送付先に反映させます (fire and forget とも呼ばれます)。

qdr_accepted_deliveries_total

ルーターで受け入れられる配信の総数。受け入れられる配信についての詳細は、Understanding message routing を参照してください。

qdr_released_deliveries_total

ルーターでリリースされる配信の総数。リリースされる配信についての詳細は、Understanding message routing を参照してください。

qdr_rejected_deliveries_total

ルーターで拒否される配信の総数。拒否される配信についての詳細は、Understanding message routing を参照してください。

qdr_modified_deliveries_total

ルーターで変更される配信の総数。変更された配信についての詳細は、Understanding message routing を参照してください。

qdr_deliveries_ingress_total

クライアントからルーターに配信されるメッセージの総数。これには管理メッセージが含まれますが、ルート制御メッセージは含まれません。

qdr_deliveries_egress_total

ルーターからクライアントに送信されるメッセージの総数。これには管理メッセージが含まれますが、ルート制御メッセージは含まれません。

qdr_deliveries_transit_total, qdr_deliveries_ingress_route_container_total

別のルーターに配信するためにルーターを通過するメッセージの総数。

qdr_deliveries_egress_route_container_total

ルーターから AMQP ルートコンテナーに送信された配信の総数。これには、AMQ Broker インスタンスへのメッセージおよび管理メッセージが含まれますが、ルート制御メッセージは含まれません。

qdr_deliveries_delayed_1sec_total

ルーターによって転送され 1 秒間処理されなかった配信の総数。

qdr_deliveries_delayed_10sec_total

ルーターによって転送され 10 秒間処理されなかった配信の総数。

qdr_deliveries_stuck_total

配信できない配信の総数。Message routing flow control で説明されているように、通常、配信は認証情報がないため配信することができません。

qdr_links_blocked_total

ブロックされるリンクの総数。

qdr_deliveries_redirected_to_fallback_total

フォールバック宛先に転送された配信の総数。詳細は、Handling undeliverable messages を参照してください。

関連情報

「Grafana での AMQ Interconnect ダッシュボードの表示」を参照してください。

第10章 AMQ INTERCONNECT WEB コンソールを使用したルーターネットワークの監視

AMQ Interconnect Web コンソールを使用して、ルーターネットワークのステータスとパフォーマンスを監視できます。デフォルトでは、ルーターデプロイメントを作成すると、AMQ Interconnect Operator はコンソールにアクセスするための認証情報を生成し、それらをシークレットに保存します。

手順

1. OpenShift で **Networking → Routes** に切り替えて、コンソール Route をクリックします。Web コンソールが新規タブで開きます。
2. Web コンソールに接続するには、以下のフィールドを入力します。

   Port

   443 と入力します。

   User name

   ユーザー名を入力します。
   Web コンソールにアクセスするためのユーザー名およびパスワードを見つけるには、**Workloads → Secrets** に移動します。Web コンソールの認証情報を含むシークレットは **<application-name>-users** (例: **router-mesh-users**) と呼ばれます。

   ユーザー名の構文は <**user**>@<**domain**> (domain は OpenShift アプリケーションの名前で、ルーターデプロイメントを記述するカスタムリソースの名前です) です (例: **guest@router-mesh**)。

   Password

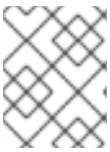
   <application-name>-users シークレットで定義したパスワードを入力します。
3. **Connect** をクリックします。
   Routers ページが表示され、ルーターネットワーク内のすべてのルーターが表示されます。
4. Web コンソールのタブを使用してルーターのネットワークを監視します。

タブ	提供する内容
Overview	ルーター、アドレス、リンク、接続、およびログに関する情報を集約します。
Entities	ルーターネットワーク内の各ルーターの各 AMQP 管理エンティティーに関する詳細情報。属性によっては、**Charts** タブに追加できるチャートがあります。
Topology	ルーター、クライアント、ブローカーなど、ルーターネットワークのグラフィカルビュー。トポロジーでは、ルーターの接続状態と、メッセージがネットワークを通過する様子を示します。
Charts	**Entities** タブで選択した情報のグラフ。

タブ	提供する内容
Message Flow	アドレス別にリアルタイムのメッセージフローを表示するコードダイアグラム。
Schema	ルーターネットワークの各ルーターを制御する管理スキーマ。

第11章 RED HAT INTEGRATION - AMQ CERTIFICATE MANAGER OPERATOR からの移行

Red Hat Integration - AMQ Certificate Manager Operator は接続のセキュリティー保護に必要なく、OpenShift Container Platform 4.9 ではサポートされません。OpenShift Container Platform 4.9 の接続をセキュアにするために、以下の説明に従って Red Hat Integration - AMQ Certificate Manager Operator を削除できます。



注記

Red Hat Integration - AMQ Certificate Manager Operator を削除した後も、以前に発行された証明書は引き続き有効であり、接続もセキュリティー保護されます。

前提条件

- **cluster-admin** アカウントを使用してクラスターにログインしている。

手順

1. Red Hat Integration - AMQ Interconnect Operator が最新バージョンにアップグレードされていることを確認してください。
2. Red Hat Integration - AMQ Certificate Manager Operator を削除します。
   a. OpenShift Container Platform の Web コンソールで、**Operators → OperatorHub** に移動します。
   b. インストールされている Operator のリストから **Red Hat Integration - AMQ Certificate Manager Operator** を選択し、**Remove** をクリックします。
   c. 次のメッセージが表示されたら、**OK** をクリックします。

   ```
   Operator Red Hat Integration - AMQ Certificate Manager will be removed from all-
   namespaces, but any of its custom resource definitions
   or managed resources will remain. If your Operator deployed applications on the cluster
   or configured off-cluster resources, these will
   continue to run and require manual cleanup.
   ```

3. Red Hat Integration - AMQ Certificate Manager Operator CRD を削除します。

   ```
   $ oc delete crd issuers.certmanager.k8s.io

   customresourcedefinition.apiextensions.k8s.io "issuers.certmanager.k8s.io" deleted
   ```

4. 現在のすべての接続が機能していることを確認します。

注記

Red Hat Integration - AMQ Certificate Manager Operator を削除した後に新しい接続を作成する場合は、新しい接続をセキュリティー保護するために 「SSL/TLS 認証用のシークレットの作成」 で説明されている手順に従う必要があります。

改訂日時: 2023-01-07 11:27:39 +1000