Red Hat AMQ 7.7

OpenShift への AMQ Interconnect のデプロイ

AMQ Interconnect 1.8 向け

Last Updated: 2023-02-03

Red Hat AMQ 7.7 OpenShift への AMQ Interconnect のデプロイ

AMQ Interconnect 1.8 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

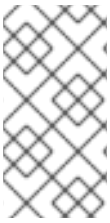
OpenShift Container Platform に AMQ Interconnect をインストールし、デプロイする方法を説明します。

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。



注記

AMQ Interconnect Operator を使用できない場合は、OpenShift カタログで提供される AMQ Interconnect アプリケーションテンプレートを使用して、OpenShift に AMQ Interconnect をデプロイできます。詳細は、OpenShift での AMQ Interconnect のデプロイ を参照してください。

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 AMQ INTERCONNECT OPERATOR のインストール

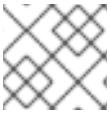
OperatorHub を使用して、以下の Operator を OpenShift Container Platform クラスターに追加します。

AMQ Certificate Manager Operator

AMQ Interconnect Operator によって使用される TLS 証明書を生成する Kubernetes アドオン。この Operator は OpenShift Container Platform クラスター用に 1 回インストールする必要があります。インストールすると、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。

AMQ Interconnect Operator

AMQ Interconnect ルーターネットワークをデプロイするための Operator。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。



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。

2.1. AMQ CERTIFICATE MANAGER OPERATOR の追加

AMQ Certificate Manager Operator (cert-manager) は、TLS 証明書を発行および管理する 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。

AMQ Certificate Manager Operator は、OpenShift Container Platform クラスター用に 1 回インストールする必要があります。インストールすると、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** アカウントを使用して OpenShift Container Platform 4.1 クラスターにアクセスできること。

手順

1. OpenShift Container Platform Web コンソールで、**Catalog → OperatorHub** に移動します。
2. 利用可能な Operator の一覧から **AMQ Certificate Manager Operator** を選択し、**Install** をクリックします。
3. **Create Operator Subscription** ページですべてのデフォルトを受け入れ、**Subscribe** をクリックします。
   これにより、Operator はこの OpenShift クラスターを使用する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。**Subscription Overview** 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. **Catalog → Installed Operators** ページに切り替え、**openshift-operators** プロジェクトに切り替えます。
   AMQ Certificate Manager Operator は、InstallSucceeded のステータスで表示されます。
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. **Catalog → Operator Management**ページに切り替え、**Operator Subscriptions** および **Install Plans** タブで **Status** の失敗またはエラーの有無を確認します。

b. **Workloads → Pods** ページに切り替え、問題を報告している Pod のログを確認します。

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。

2.2. AMQ INTERCONNECT OPERATOR の追加

AMQ Interconnect Operator は、OpenShift Container Platform で 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。

前提条件

- **cluster-admin** アカウントを使用して OpenShift Container Platform 4.1 クラスターにアクセスできること。
- 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こと。

手順

1. OpenShift Container Platform Web コンソールで、**Catalog → OperatorHub** に移動します。
2. 利用可能な Operator の一覧から **AMQ Interconnect Operator** を選択し、**Install** をクリックします。
3. **Create Operator Subscription** ページで、Operator をインストールする namespace を選択し、**Subscribe** をクリックします。
   Subscription Overview 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. **Catalog → Installed Operators** ページに切り替え、AMQ Interconnect Operator が表示され、その **Status** が **InstallSucceeded** であることを確認します。
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. **Catalog → Operator Management**ページに切り替え、**Operator Subscriptions** および **Install Plans** タブで **Status** の失敗またはエラーの有無を確認します。
   b. **Workloads → Pods** ページに切り替え、問題を報告している Pod のログを確認します。

第3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。

3.1. 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。

Anti-Affinity

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. AMQ Certificate Manager Operator を使用すると、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。

3.2. 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
```

3.3. 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
```

   AMQ Interconnect Operator は、デフォルト設定で内部ルーターを作成します。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
---
```

(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 --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 --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
    caCert: cluster2-inter-router-tls
  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4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、AMQ Interconnect Operator はルーターデプロイメントのサービスを作成し、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。

4.1. 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
  ...
```

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に接続するために使用できるルートを作成します。

4.2. クライアント接続の認証

ルーターデプロイメントの作成時に、AMQ Interconnect Operator は AMQ Certificate Manager Operator を使用してクライアント認証用のデフォルトの 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。

4.3. 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5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、リンクルートの作成 を参照してください。

第6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。

6.1. 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の詳細は、メッセージルートの設定 を参照してください。

6.2. 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)
  - prefix: my-queue
    direction: in
    connection: my-broker
  - prefix: 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

- ブローカーキューとの間でメッセージのルーティングに関する詳細は、ブローカーキューを使用したメッセージのルーティング を参照してください。

第7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。

7.1. 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 設定に必要な設定ファイルが含まれています。

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

2. **deploy-monitoring.sh** スクリプトを開き、**NAMESPACE** 変数を設定します。
NAMESPACE 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。

```
#!/bin/bash

# Change the namespace to that of your project
NAMESPACE=myproject
...
```

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
```

4. prometheus、alertmanager、および grafana サービスのルートを作成します。

```
$ oc expose service prometheus

$ oc expose service alertmanager

$ oc expose service grafana
```

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。

7.2. 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。

- **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**

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7.1 Delayed Deliveries ダッシュボード

第8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。

改訂日時：2023-01-28 11:51:04 +1000