Red Hat AMQ 7.7

OpenShift での AMQ Online のインストールと管理

AMQ Online 1.5 で使用する場合

Last Updated: 2023-02-28

Red Hat AMQ 7.7 OpenShift での AMQ Online のインストールと管理

AMQ Online 1.5 で使用する場合

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

このガイドでは、AMQ Online をインストールして管理する方法について説明します。

目次

第1章 はじめに

1.1. AMQ ONLINE の概要

Red Hat AMQ Online は、マネージドサービスとしてメッセージングを配信するための OpenShift ベースのメカニズムです。Red Hat AMQ Online を使用すると、管理者は、クラウドまたはオンプレミスのいずれかで、クラウドネイティブのマルチテナントメッセージングサービスを設定できます。開発者は、Red Hat AMQ コンソールを使用してメッセージングをプロビジョニングできます。複数の開発チームがコンソールからブローカーとキューをプロビジョニングできます。各チームがソフトウェアをインストール、設定、デプロイメント、保守、またはパッチを適用する必要はありません。

AMQ Online は、ユースケースに応じてさまざまな種類のメッセージングをプロビジョニングできます。ユーザーは、Address Space を作成することでメッセージングリソースをリクエストできます。AMQ Online は現在、標準とブローカーの 2 つのアドレス空間タイプをサポートしており、それぞれセマンティクスが異なります。次の図は、各アドレス空間タイプのアーキテクチャーの概要を示しています。

図1.1 標準アドレス空間

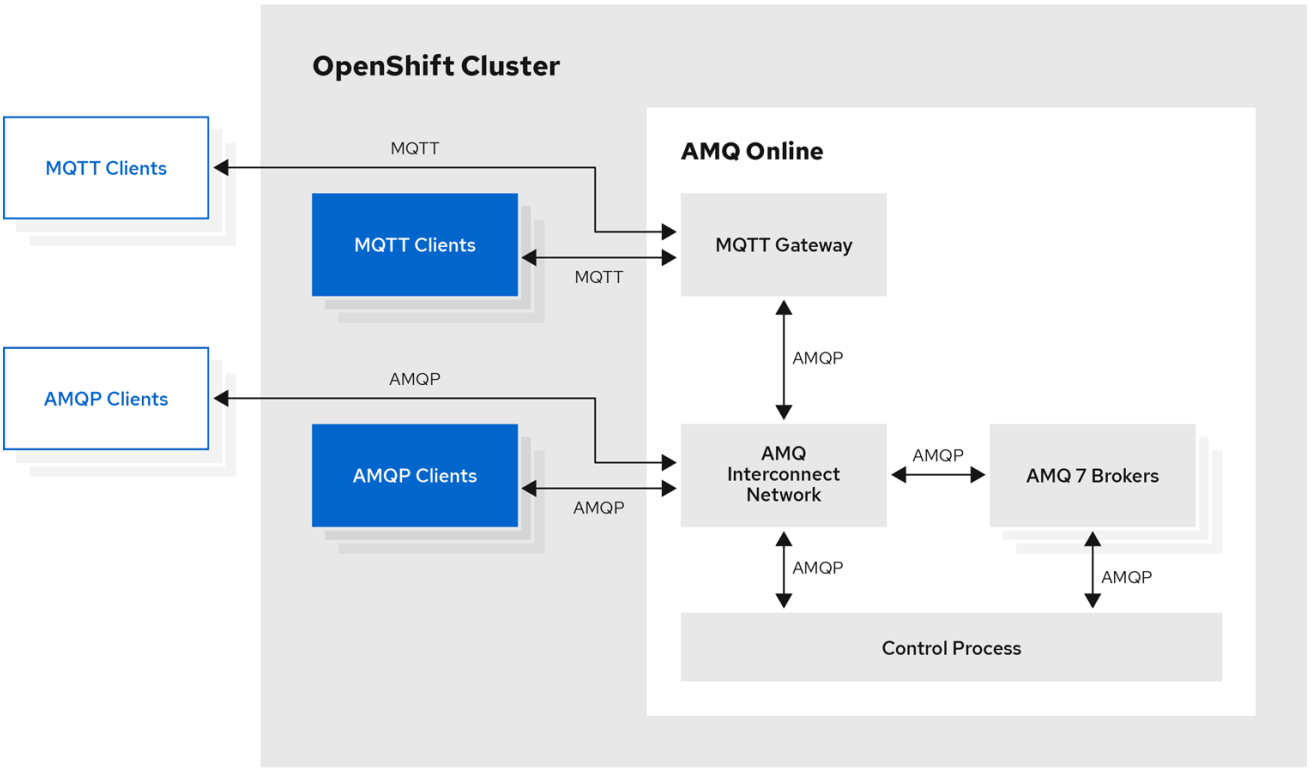
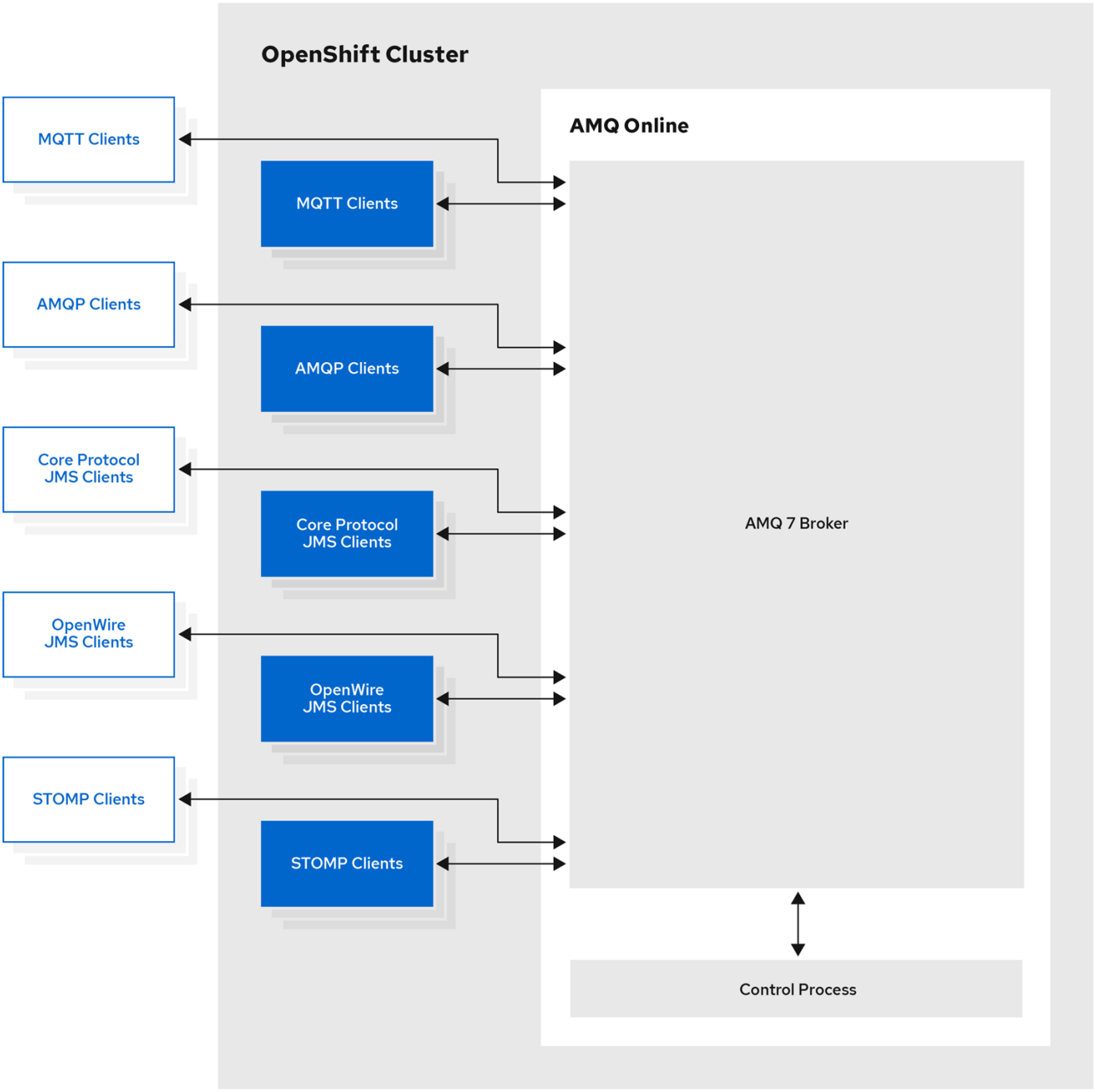


図1.2 ブローカーアドレス空間



1.2. サポートされる機能

次の表は、AMQ Online 1.5 でサポートされている機能を示しています。

表1.1 サポート対象機能の参照表

機能		ブローカーアドレス空間	標準アドレス空間
アドレスの種類	Queue	はい	はい
	トピック	はい	はい

機能		ブローカーアドレス空間	標準アドレス空間
	マルチキャスト	いいえ	はい
	anycast	いいえ	はい
	Subscription	いいえ	はい
メッセージングプロトコル	AMQP	はい	はい
	MQTT	はい	テクノロジープレビューとしてのみ提供
	CORE	はい	いいえ
	OpenWire	はい	いいえ
	STOMP	はい	いいえ
トランスポート	TCP	はい	はい
	WebSocket	はい	はい
永続サブスクリプション	JMS 永続サブスクリプション	はい	いいえ
	名前付き永続サブスクリプション	いいえ	はい
JMS	トランザクションサポート	はい	いいえ
	キューのセレクター	はい	いいえ
	メッセージ順序の保証 (優先順位付けを含む)	はい	いいえ
スケーラビリティー	スケーラブルな分散キューとトピック	いいえ	はい

1.3. AMQ ONLINE ユーザーのロール

AMQ Online ユーザーは、サービス管理者とメッセージングテナントの 2 つのユーザーロールに関して広く定義できます。組織の規模に応じて、これらのロールは同じユーザーまたは異なるユーザーによって実行される場合があります。

サービス管理者は、初期インストールとその後のアップグレードを実行します。サービス管理者は、ルーター、ブローカー、および管理コンポーネントの監視など、メッセージングインフラストラクチャーをデプロイメントおよび管理するだけでなく、アドレス空間計画やアドレス計画も作成しま

す。**OpenShift での AMQ Online のインストールと管理** では、AMQ Online をセットアップおよび管理する方法と、サービス管理者としてインフラストラクチャーとプランを設定する方法について説明しています。

メッセージングテナントは、クラウドネイティブの API とツールの両方を使用して、メッセージングリソースを要求できます。メッセージングテナントは、メッセージングシステム内の特定のアドレス空間のユーザーとアクセス許可を管理したり、アドレス管理とアドレスを作成したりすることもできます。アドレス空間、アドレス、およびユーザーを管理する方法の詳細は **OpenShift Container Platform での AMQ Online の使用** を参照してください。

1.4. サポートされる構成

AMQ Online でサポートされる設定の詳細は、Red Hat AMQ 7 でのサポート対象設定 を参照してください。

1.5. 本書の表記慣例

1.5.1. 変数テキスト

本書では、変数を含むコードブロックが紹介されていますが、これは、お客様のシステム環境に固有の値に置き換える必要があります。このドキュメントでは、そのようなテキストはイタリックモノスペースのスタイルで指定しています。

たとえば、次のコードブロックで、**my-namespace** はインストールで使用されている namespace に置き換えます。

```
sed -i 's/amq-online-infra/my-namespace/' install/bundles/enmasse-with-standard-authservice/*.yaml
```

第2章 AMQ ONLINE のインストール

AMQ Online は、OpenShift Container Platform コマンドラインインターフェイスを使用して YAML ファイルを適用するか、Ansible Playbook を実行してインストールできます。

前提条件

- Debezium をインストールするには、OpenShift Container Platform コマンドラインインターフェイス (CLI) が必要です。
  - OpenShift 3.x の CLI のインストール方法については、OpenShift Container Platform 3.11 のドキュメント を参照してください。
  - OpenShift 4.1 の CLI のインストール方法については、OpenShift Container Platform 4.1 のドキュメント を参照してください。
- OpenShift クラスターが必要です。
- 必要なクラスターロールおよび API サービスを設定するための **cluster-admin** 権限を持つ OpenShift クラスターのユーザーが必要です。

2.1. AMQ ONLINE のダウンロード

手順

- AMQ Online ダウンロードサイト から **amq-online-install.zip** ファイルをダウンロードして展開します。

注記

AMQ Online のコンテナーイメージは Red Hat Container Catalog で使用できますが、この代わりに提供される YAML ファイルを使用することが推奨されます。

2.2. YAML バンドルを使用した AMQ ONLINE のインストール

AMQ Online をインストールする最も簡単な方法は、定義済みの YAML バンドルを使用することです。

手順

1. **cluster-admin** 権限を持つユーザーとしてログインしてます。

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

2. (オプション) **amq-online-infra** 以外のプロジェクトにデプロイする場合は、次のコマンドを実行し、後続の手順で **amq-online-infra** を置き換える必要があります。

```
sed -i 's/amq-online-infra/my-project/' install/bundles/amq-online/*.yaml
```

3. AMQ Online をデプロイするプロジェクトを作成します。

```
oc new-project amq-online-infra
```

4. ダウンロードしたリリースファイルの場所にディレクトリーを変更します。

5. **amq-online** バンドルを使用してデプロイします。

```
oc apply -f install/bundles/amq-online
```

6. (オプション) サンプルプランとインフラストラクチャー設定をインストールします。

```
oc apply -f install/components/example-plans
```

7. (オプション) サンプルのロールをインストールします。

```
oc apply -f install/components/example-roles
```

8. (オプション) **標準** 認証サービスをインストールします。

```
oc apply -f install/components/example-authservices/standard-authservice.yaml
```

9. (オプション) Service Catalog 統合をインストールします。

```
oc apply -f install/components/service-broker
oc apply -f install/components/cluster-service-broker
```

2.3. ANSIBLE を使用した AMQ ONLINE のインストール

Ansible を使用して AMQ Online をインストールするには、システム設定の変数を含むインベントリーファイルを作成する必要があります。インベントリーファイルの例は **ansible/inventory** フォルダーにあります。

次のインベントリーファイルの例では、AMQ Online の最小限のインストールを有効にします。

```
[enmasse]
localhost ansible_connection=local

[enmasse:vars]
namespace=amq-online-infra
enable_rbac=False
api_server=True
service_catalog=False
register_api_server=True
keycloak_admin_password=admin
authentication_services=["standard"]
standard_authentication_service_postgresql=False
monitoring_namespace=enmasse-monitoring
monitoring_operator=False
monitoring=False
```

次の Ansible 設定設定がサポートされています。

表2.1 Ansible 構成設定

名前	説明	デフォルト値	Required
namespace	AMQ Online がインストールされているプロジェクトを指定します。	該当なし	はい
enable_rbac	REST API の RBAC 認証を有効にするかどうかを指定します	True	いいえ
service_catalog	Service Catalog との統合を有効にするかどうかを指定します	False	いいえ
authentication_services	デプロイする認証サービスのリストを指定します。サポートされている値は **none** と **standard** です。	**none**	いいえ
keycloak_admin_password	**standard** 認証サービス Red Hat Single Sign-On インスタンスに使用する管理者パスワードを指定します	該当なし	はい (**標準** 認証サービスが有効な場合)
api_server	REST API サーバーを有効にするかどうかを指定します	True	いいえ
register_api_server	API サーバーを OpenShift マスターに登録するかどうかを指定します	False	いいえ
secure_api_server	API サーバーの相互 TLS を有効にするかどうかを指定します	False	いいえ
install_example_plans	サンプルプランとインフラストラクチャー設定をインストールするかどうかを指定します	True	いいえ
monitoring_namespace	AMQ Online 監視がインストールされているプロジェクトを指定します。	該当なし	はい
monitoring_operator	監視インフラストラクチャーをインストールするかどうかを指定します	該当なし	いいえ

手順

1. インベントリーファイルを作成します。
2. Ansible Playbook の実行:

```
ansible-playbook -i inventory-file ansible/playbooks/openshift/deploy_all.yml
```

2.4. OPERATOR LIFECYCLE MANAGER を使用した AMQ ONLINE のインストールと設定

Operator Lifecycle Manager を使用して、AMQ Online のインスタンスをインストールおよび設定できます。

OpenShift 4x 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、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 4.x 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、アクセスをこれに付与するのに役立ちます。OpenShift コンソールは、クラスター管理者が Operator をインストールしたり、クラスターで利用可能な Operator のカタログを使用できるように特定のプロジェクトアクセスを付与したりするのに使用する管理画面を提供します。

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。

2.4.1. OpenShift コンソールを使用した OperatorHub から AMQ Online のインストール

OpenShift コンソールで OperatorHub を使用して、AMQ Online Operator を OpenShift 4.x クラスターにインストールできます。

前提条件

- OpenShift 4.x クラスターへのアクセス権限と、**クラスター管理者** 権限を持つアカウント。

手順

1. OpenShift 4.x コンソールで、**cluster-admin** 権限を持つアカウントを使用してログインします。
2. AMQ Online をデプロイするプロジェクトを作成するには、**Home > Projects** をクリックし、**Create Project** をクリックします。プロジェクトの作成ウィンドウが開きます。
3. **Name** フィールドに **amq-online-infra** と入力し、**Create** をクリックします。**amq-online-infra** プロジェクトが作成されます。
4. **Operators > OperatorHub** の順にクリックします。
5. **Filter by keyword** ボックスに **AMQ Online** と入力して、AMQ Online Operator を見つけます。
6. AMQ Online Operator をクリックします。operator に関する情報が表示されます。

7. Operator についての情報を確認してから、**Install** をクリックします。Create Operator Subscription のページが表示されます。

8. **Create Operator Subscription** ページの **Installation Mode** で、**A specific namespace on the cluster** をクリックし、ドロップダウンリストから **amq-online-infra** namespace を選択します。

9. 残りのデフォルトの選択内容をすべて受け入れて、**Subscribe** をクリックします。**amq-online** ページが表示され、AMQ Online Operator サブスクリプションのインストールの進行状況を監視できます。

10. サブスクリプションのアップグレードステータスが **Up to date** と表示されたら、**Operators** > **Installed Operators** をクリックして、**AMQ Online** ClusterServiceVersion (CSV) が表示され、その **ステータス** が最終的に **amq-online-infra** namespace で **InstallSucceeded** に解決されることを確認します。
トラブルシューティングに関する情報は、OpenShift のドキュメント を参照してください。

次のステップ

- OpenShift コンソールを使用した AMQ Online の設定

2.4.2. OpenShift コンソールを使用した AMQ Online の設定

OpenShift コンソールを使用して OperatorHub から AMQ Online をインストールした後、**amq-online-infra** プロジェクト内の以下の項目に対してカスタムリソースの新しいインスタンスを作成します。

- 認証サービス
- アドレス空間タイプのインフラストラクチャー設定 (例では標準のアドレス空間タイプを使用)
- アドレス空間計画
- アドレス計画

カスタムリソースの新しいインスタンスを作成したら、次に以下を実行します。

- アドレス空間の作成
- アドレスの作成
- メッセージングユーザーの作成

以下の手順では、OpenShift コンソールの使用時に提供されるサンプルデータを使用します。

2.4.2.1. OpenShift コンソールを使用した認証サービスのカスタムリソースの作成

AMQ Online を使用するには、認証サービス用のカスタムリソースを作成する必要があります。この例では、標準の認証サービスを使用しています。

手順

1. 右上にある **プラス** アイコン (+) をクリックします。YAML のインポートウィンドウが開きます。

2. 左上のドロップダウンメニューから、**amq-online-infra** プロジェクトを選択します。

3. 次のコードをコピーします。

```
apiVersion: admin.enmasse.io/v1beta1
kind: AuthenticationService
metadata:
  name: standard-authservice
spec:
  type: standard
```

4. YAML のインポートウィンドウで、コピーしたコードを貼り付けて **作成** をクリックします。AuthenticationService の概要ページが表示されます。

5. **Workloads > Pods**をクリックします。カスタムリソースがデプロイされると、**Readiness** 列の Pod ステータスは **Ready** になります。

次のステップ

- OpenShift コンソールを使用したインフラストラクチャー設定のカスタムリソース作成

2.4.2.2. OpenShift コンソールを使用したインフラストラクチャー設定のカスタムリソース作成

AMQ Online を使用するには、インフラストラクチャー設定のカスタムリソースを作成する必要があります。この例では、標準アドレス空間に **StandardInfraConfig** を使用します。

手順

1. 右上にある **プラス** アイコン (+) をクリックします。YAML のインポートウィンドウが開きます。

2. 左上のドロップダウンメニューから、**amq-online-infra** プロジェクトを選択します。

3. 次のコードをコピーします。

```
apiVersion: admin.enmasse.io/v1beta1
kind: StandardInfraConfig
metadata:
  name: default
```

4. YAML のインポートウィンドウで、コピーしたコードを貼り付けて **作成** をクリックします。StandardInfraConfig の概要ページが表示されます。

5. **Operators > Installed Operators**の順にクリックします。

6. AMQ Online Operator をクリックし、**Standard Infra Config** タブをクリックして、**Status** が **Active** と表示されていることを確認します。

次のステップ

- OpenShift コンソールを使用したアドレス空間計画のカスタムリソースの作成

2.4.2.3. OpenShift コンソールを使用したアドレス空間計画のカスタムリソースの作成

AMQ Online を使用するには、アドレス空間計画のカスタムリソースを作成する必要があります。この手順では、OpenShift コンソールの使用時に提供されるサンプルデータを使用します。

手順

1. 右上にある **プラス** アイコン (+) をクリックします。YAML のインポートウィンドウが開きます。
2. 左上のドロップダウンメニューから、**amq-online-infra** プロジェクトを選択します。
3. 次のコードをコピーします。

```
apiVersion: admin.enmasse.io/v1beta2
kind: AddressSpacePlan
metadata:
  name: standard-small
spec:
  addressSpaceType: standard
  infraConfigRef: default
  addressPlans:
    - standard-small-queue
  resourceLimits:
    router: 2.0
    broker: 3.0
    aggregate: 4.0
```

4. YAML のインポートウィンドウで、コピーしたコードを貼り付けて **作成** をクリックします。AddressSpacePlan の概要ページが表示されます。
5. **Operators > Installed Operators**の順にクリックします。
6. AMQ Online Operator をクリックし、**Address Space Plan** タブをクリックして、**Status** が **Active** と表示されていることを確認します。

次のステップ

- OpenShift コンソールを使用したアドレス計画のカスタムリソース作成

2.4.2.4. OpenShift コンソールを使用したアドレス計画カスタムリソースの作成

AMQ Online を使用するには、アドレス計画のカスタムリソースを作成する必要があります。この手順では、OpenShift コンソールの使用時に提供されるサンプルデータを使用します。

手順

1. 右上にある **プラス** アイコン (+) をクリックします。YAML のインポートウィンドウが開きます。
2. 左上のドロップダウンメニューから、**amq-online-infra** プロジェクトを選択します。
3. 次のコードをコピーします。

```
apiVersion: admin.enmasse.io/v1beta2
kind: AddressPlan
metadata:
```

```
  name: standard-small-queue
spec:
  addressType: queue
  resources:
    router: 0.01
    broker: 0.1
```

4. YAML のインポートウィンドウで、コピーしたコードを貼り付けて **作成** をクリックします。AddressPlan の概要ページが表示されます。

5. **Operators > Installed Operators**の順にクリックします。

6. AMQ Online Operator をクリックし、**Address Plan** タブをクリックして、**Status** が **Active** と表示されていることを確認します。

次のステップ

- アドレス空間の作成
- アドレスの作成
- メッセージングユーザーの作成

第3章 AMQ ONLINE のアップグレード

AMQ Online は、クラウドネイティブツールを使用したマイナーバージョン間のアップグレードをサポートしています。アップグレード時に、設定の変更を適用すると、アップグレードプロセスが自動的に開始されます。

AMQ Online の最初のインストールに使用したのと同じ方法を使用して、AMQ Online の新しいバージョンにアップグレードすることをお勧めします。

AMQ Online のアップグレードは、新しいバージョンの YAML ファイルを適用することによって実行されます。

3.1. YAML バンドルを使用した AMQ ONLINE のアップグレード

前提条件

- AMQ Online の新しいリリース。詳細は、AMQ Online のダウンロード を参照してください。

手順

1. サービスオペレーターとしてログインします。

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

2. AMQ Online がインストールされているプロジェクトを選択します。

```
oc project amq-online-infra
```

3. 新しいリリースバンドルを適用します。

```
oc apply -f install/bundles/amq-online
```

4. 再起動中に Pod を監視します。

```
oc get pods -w
```

Pod が再起動し、数分以内にアクティブになります。

5. アップグレード後に不要になった **api-server** リソースを削除します。

```
oc delete sa api-server -n amq-online-infra
oc delete clusterrolebinding enmasse.io:api-server-amq-online-infra
oc delete clusterrole enmasse.io:api-server
oc delete rolebinding api-server -n amq-online-infra
oc delete role enmasse.io:api-server -n amq-online-infra
```

3.2. ANSIBLE を使用した AMQ ONLINE のアップグレード

前提条件

- AMQ Online の新しいリリース。詳細は、AMQ Online のダウンロード を参照してください。

手順

1. サービスオペレーターとしてログインします。

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

2. 新しいリリースから Ansible Playbook を実行します。

```
ansible-playbook -i inventory-file ansible/playbooks/openshift/deploy_all.yml
```

3. 再起動中に Pod を監視します。

```
oc get pods -w
```

Pod が再起動し、数分以内にアクティブになります。

4. アップグレード後に不要になった **api-server** リソースを削除します。

```
oc delete sa api-server -n amq-online-infra
oc delete clusterrolebinding enmasse.io:api-server-amq-online-infra
oc delete clusterrole enmasse.io:api-server
oc delete rolebinding api-server -n amq-online-infra
oc delete role enmasse.io:api-server -n amq-online-infra
```

第4章 AMQ ONLINE のインストール

AMQ Online のインストールに使用したのと同じ方法を使用して、AMQ Online をアンインストールする必要があります。

4.1. YAML バンドルを使用した AMQ ONLINE のアンインストール

このメソッドは、YAML バンドルを使用してインストールされた AMQ Online をアンインストールします。

手順

1. **cluster-admin** 権限を持つユーザーとしてログインしてます。

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

2. クラスターレベルのリソースを削除します。

```
oc delete crd -l app=enmasse,enmasse-component=iot
oc delete crd -l app=enmasse --timeout=600s
oc delete clusterrolebindings -l app=enmasse
oc delete clusterroles -l app=enmasse
oc delete apiservices -l app=enmasse
oc delete oauthclients -l app=enmasse
```

3. (OpenShift 4) コンソール統合を削除します。

```
oc delete consolelinks -l app=enmasse
```

4. (オプション) サービスカタログ統合を削除します。

```
oc delete clusterservicebrokers -l app=enmasse
```

5. AMQ Online がデプロイされているプロジェクトを削除します。

```
oc delete project amq-online-infra
```

4.2. ANSIBLE を使用した AMQ ONLINE のアンインストール

Ansible を使用して AMQ Online をアンインストールするには、AMQ Online のインストールに使用したものと同じインベントリーファイルを使用する必要があります。

注記

Playbook は **amq-online-infra** プロジェクトを削除します。

手順

1. Ansible Playbook を実行します。ここで、**inventory-file** は、インストール時に使用されるインベントリーファイルを指定します。

```
ansible-playbook -i inventory-file ansible/playbooks/openshift/uninstall.yml
```

4.3. OPERATOR LIFECYCLE MANAGER (OLM) を使用した AMQ ONLINE のアンインストール

このメソッドは、Operator Lifecycle Manager (OLM) を使用してインストールされた AMQ Online をアンインストールします。

手順

1. **cluster-admin** 権限を持つユーザーとしてログインしてます。

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

2. すべての **IoTProject** および **AddressSpace** インスタンスを削除します。

```
oc delete iotprojects -A --all
oc delete addressspaces -A --all --timeout=600s
```

3. サブスクリプションを削除します (**amq-online** をインストールで使用したサブスクリプションの名前に置き換えます)。

```
oc delete subscription amq-online -n amq-online-infra
```

4. Operator の CSV を削除します。

```
oc delete csv -l app=enmasse -n amq-online-infra
```

5. 残りのリソースをすべて削除します (**amq-online-infra** は、AMQ Online をインストールしたプロジェクトに置き換えます)。

```
oc delete all -l app=enmasse -n amq-online-infra
oc delete cm -l app=enmasse -n amq-online-infra
oc delete secret -l app=enmasse amq-online-infra
oc delete consolelinks -l app=enmasse
oc delete oauthclients -l app=enmasse
oc delete crd -l app=enmasse
```

6. (オプション: AMQ Online が **openshift-operators** namespace にインストールされている場合は、この手順をスキップします) AMQ Online がインストールされた namespace を削除します。

```
oc delete namespace amq-online-infra
```

4.4. OPENSHIFT コンソールを使用した AMQ ONLINE のアンインストール

このメソッドは、OpenShift Container Platform コンソールで Operator Lifecycle Manager (OLM) を使用してインストールされた AMQ Online をアンインストールします。

手順

1. プロジェクトリストから、AMQ Online をインストールしたプロジェクトを選択します。
2. **Catalog → Operator Management** をクリックします。Operator 管理ページが開きます。
3. **Operator Subscriptions** タブをクリックします。
4. アンインストールする AMQ Online Operator を見つけます。右端の列で、縦の省略記号アイコンをクリックし、**Remove Subscription** を選択します。
5. Remove Subscription ウィンドウでプロンプトが表示されたら、Operator に関連するすべてのコンポーネントを削除する場合は、**Also completely remove the AMQ Online Operator from the selected namespace** チェックボックスを選択してください。
   - **Remove** をクリックします。AMQ Online Operator は実行を停止し、更新を受信しなくなります。
6. 次のコマンドを実行して、残りのリソースをすべて削除します (**amq-online-infra** は、AMQ Online をインストールしたプロジェクトに置き換えます)。

```
oc delete all -l app=enmasse -n amq-online-infra
oc delete cm -l app=enmasse -n amq-online-infra
oc delete secret -l app=enmasse amq-online-infra
oc delete consolelinks -l app=enmasse
oc delete oauthclients -l app=enmasse
```

7. (オプション: AMQ Online が **openshift-operators** namespace にインストールされている場合は、この手順をスキップします) AMQ Online がインストールされた namespace を削除します。

```
oc delete namespace amq-online-infra
```

第5章 AMQ ONLINE の設定

5.1. 最小限のサービス設定

AMQ Online を実稼働用に設定するには、ある程度の時間と考慮が必要です。次の手順では、最小限のサービス設定から始めます。より完全な例については、AMQ Online ディストリビューションの **install/components/example-plans** フォルダーに移動してください。すべてのコマンドは、AMQ Online がインストールされている namespace で実行する必要があります。

手順

1. 設定例を保存します。

```
apiVersion: admin.enmasse.io/v1beta1
kind: StandardInfraConfig
metadata:
  name: default
spec: {}
---
apiVersion: admin.enmasse.io/v1beta2
kind: AddressPlan
metadata:
  name: standard-small-queue
spec:
  addressType: queue
  resources:
    router: 0.01
    broker: 0.1
---
apiVersion: admin.enmasse.io/v1beta2
kind: AddressSpacePlan
metadata:
  name: standard-small
spec:
  addressSpaceType: standard
  infraConfigRef: default
  addressPlans:
  - standard-small-queue
  resourceLimits:
    router: 2.0
    broker: 3.0
    aggregate: 4.0
---
apiVersion: admin.enmasse.io/v1beta1
kind: AuthenticationService
metadata:
  name: none-authservice
spec:
  type: none
```

2. 設定例を適用します。

```
oc apply -f service-config.yaml
```

5.2. アドレス空間計画

アドレス空間計画は、クォータを設定し、アドレス空間によって消費されるリソースを制御するために使用されます。アドレス空間計画は AMQ Online サービス Operator により設定され、アドレス空間の作成時にメッセージングテナントにより選択されます。

AMQ Online には、ほとんどのユースケースに十分対応できるデフォルトの計画セットが含まれています。

計画はカスタムリソースとして設定されます。次の例は、標準アドレス空間の計画を示しています。

```
apiVersion: admin.enmasse.io/v1beta2
kind: AddressSpacePlan
metadata:
  name: restrictive-plan
  labels:
    app: enmasse
spec:
  displayName: Restrictive Plan
  displayOrder: 0
  infraConfigRef: default (1)
  shortDescription: A plan with restrictive quotas
  longDescription: A plan with restrictive quotas for the standard address space
  addressSpaceType: standard (2)
  addressPlans: (3)
  - small-queue
  - small-anycast
  resourceLimits: (4)
    router: 2.0
    broker: 2.0
    aggregate: 2.0
```

(1) このプランを使用してアドレス空間にデプロイされたインフラストラクチャーを記述する **StandardInfraConfig** (**standard** アドレス空間タイプの場合) または **BrokeredInfraConfig** (**brokered** アドレス空間タイプの場合) への参照。

(2) このプランが適用されるアドレス空間のタイプで、**standard** または **brokered** のいずれかです。

(3) この計画を使用するアドレス空間で使用可能なアドレス計画のリスト。

(4) このプランを使用するアドレス空間のルーター (**router**) とブローカー (**broker**) の最大数。**brokered** アドレス空間タイプの場合、**broker** フィールドのみが必要です。

その他のフィールドは、Red Hat AMQ コンソール UI によって使用されます。フィールド **spec.infraConfigRef** に注意してください。これは、このプランを使用するアドレス空間が作成されるときに存在する必要があるインフラストラクチャー設定を指します。インフラストラクチャー設定の詳細は、インフラストラクチャー設定 を参照してください。

5.3. アドレス空間計画の作成

手順

1. サービス管理者としてログインします。
   -

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

2. AMQ Online がインストールされているプロジェクトを選択します。

```
oc project amq-online-infra
```

3. アドレス空間計画定義を作成します。

```
apiVersion: admin.enmasse.io/v1beta2
kind: AddressSpacePlan
metadata:
  name: restrictive-plan
  labels:
    app: enmasse
spec:
  displayName: Restrictive Plan
  displayOrder: 0
  infraConfigRef: default
  shortDescription: A plan with restrictive quotas
  longDescription: A plan with restrictive quotas for the standard address space
  addressSpaceType: standard
  addressPlans:
  - small-queue
  - small-anycast
  resourceLimits:
    router: 2.0
    broker: 2.0
    aggregate: 2.0
```

4. アドレス空間を作成します。

```
oc create -f restrictive-plan.yaml
```

5. スキーマが更新され、計画が含まれていることを確認します。

```
oc get addressspaceschema standard -o yaml
```

5.4. アドレス計画

アドレス計画は、特定のアドレスの予想されるリソース使用量を指定します。すべてのリソースタイプのリソース使用量の合計によって、アドレス空間にプロビジョニングされるインフラストラクチャーの量が決まります。単一のルーターおよびブローカー Pod の最大使用量は 1 です。新しいアドレスが追加のリソースを必要とし、リソースの消費がアドレス空間計画の制限内である場合、増加した負荷を処理するために新しい Pod が自動的に作成されます。

アドレス計画は、AMQ Online サービス Operator によって設定され、アドレスの作成時に選択されます。

AMQ Online には、ほとんどのユースケースに十分対応できるアドレス計画のデフォルトセットが含まれています。

アドレス空間計画 セクションでは、アドレス空間計画は **small-queue** と **small-anycast** の 2 つのアドレス計画を参照します。これらのアドレス計画はカスタムリソースとして保存され、次のように定義されます。

```
apiVersion: admin.enmasse.io/v1beta2
kind: AddressPlan
metadata:
  name: small-queue
  labels:
    app: enmasse
spec:
  displayName: Small queue plan
  displayOrder: 0
  shortDescription: A plan for small queues
  longDescription: A plan for small queues that consume little resources
  addressType: queue (1)
  resources: (2)
    router: 0.2
    broker: 0.3
  partitions: 1 (3)
  messageTtl: (4)
    minimim: 30000
    maximum: 300000
```

1. この計画が適用されるアドレスタイプ。
2. この計画を使用してアドレスによって消費されたリソース。**router** フィールドは、**brokered** アドレス空間計画によって参照されるアドレス計画ではオプションです。
3. このプランを使用してキュー用に作成する必要があるパーティションの数。**standard** アドレス空間でのみ使用できます。
4. (オプション) メッセージの存続可能時間 (TTL) を制限します。アドレスタイプの **queue** と **topic** にのみ適用されます。

その他のフィールドは、Red Hat AMQ コンソール UI によって使用されます。

この計画では、単一のルーターがアドレスの 5 つのインスタンスを、ブローカーはアドレスの 3 つのインスタンスをサポートできます。この計画のアドレス数が 4 つに増えると、別のブローカーが作成されます。さらに 6 に増えると、別のルーターも作成されます。

standard アドレス空間では、**queue** アドレスタイプのアドレス計画にフィールド **partitions** が含まれる場合があります。これにより、HA とパフォーマンス向上のために複数のブローカー間でキューを分割できます。**broker** リソースの量を 1 より大きい数量に指定すると、キューが自動的に分割されます。

messageTtl フィールドは、キューまたはトピックに書き込まれたメッセージの有効な **absolute-expiry-time** を制限するために使用されます。**maximum** と **minimum** の値はミリ秒単位で定義されます。システムは、次の値に基づいて特定のアドレスへの着信メッセージの TTL 値を調整します。

- TTL 値が **maximum** の値より大きいアドレスにメッセージが到着すると、システムはメッセージの TTL を最大値に変更します。
- TTL 値が **minimum** の値より小さいアドレスにメッセージが到着すると、システムはメッセージの TTL を最小値に変更します。

TTL が定義されていない状態で到着したメッセージは、TTL 値が無限大であると見なされます。

期限切れのメッセージは、**queue**、**subscription**、または一時トピックサブスクリプションから定期的に自動削除されます。これらのメッセージは失われます。これは 30 秒ごとに発生します。

注記

シャードキューでは、メッセージの順序が保証されなくなりました。

アドレス空間計画 のアドレス空間計画の例では、2 つのルーターと 2 つのブローカーをデプロイできますが、合計で 2 つの Pod しかデプロイできません。これは、**small-queue** プランではアドレス空間が 3 つのアドレスに制限されることを意味します。

small-anycast プランはブローカーリソースを消費せず、ブローカーを作成できないという犠牲を払って 2 つのルーターをプロビジョニングできます。

```
apiVersion: admin.enmasse.io/v1beta2
kind: AddressPlan
metadata:
  name: small-anycast
  labels:
    app: enmasse
spec:
  addressType: anycast
  resources:
    router: 0.2
```

このプランでは、最大 10 個のアドレスを作成できます。

5.5. アドレス計画の作成

手順

1. サービス管理者としてログインします。

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

2. AMQ Online がインストールされているプロジェクトを選択します。

```
oc project amq-online-infra
```

3. アドレス計画定義を作成します。

```
apiVersion: admin.enmasse.io/v1beta2
kind: AddressPlan
metadata:
  name: small-anycast
  labels:
    app: enmasse
spec:
  addressType: anycast
  resources:
    router: 0.2
```

4. アドレス CR を作成します。

```
oc create -f small-anycast-plan.yaml
```

5. スキーマが更新され、計画が含まれていることを確認します。

```
oc get addressspaceschema standard -o yaml
```

5.6. インフラストラクチャー設定

AMQ Online は、ルーター、ブローカー、コンソールなどのインフラストラクチャーコンポーネントを作成します。これらのコンポーネントはシステムの実行中に設定でき、AMQ Online はコンポーネントを新しい設定で自動的に更新します。AMQ Online サービス Operator は、AMQ Online の既定のインフラストラクチャー設定を編集したり、新しい設定を作成したりできます。

インフラストラクチャー設定は、1つまたは複数のアドレス空間計画から参照できます。アドレス空間計画の詳細は、アドレス空間計画 を参照してください。

BrokeredInfraConfig および **StandardInfraConfig** リソースを使用して、**brokered** インフラストラクチャーと **standard** インフラストラクチャーの両方のインフラストラクチャー設定を管理できます。

5.6.1. 仲介インフラストラクチャー設定

BrokeredInfraConfig リソースは、**仲介** されたアドレス空間によってデプロイされたインフラストラクチャーを設定するために使用されます。アドレス空間計画は、**spec.infraConfigRef** フィールドを使用して、仲介インフラストラクチャー設定を参照します。アドレス空間計画の詳細は、アドレス空間計画 を参照してください。

利用可能な仲介インフラストラクチャー設定フィールドの詳細は、仲介インフラストラクチャー設定フィールドの表 を参照してください。

5.6.1.1. 仲介インフラストラクチャー設定例

次の仲介インフラストラクチャー設定ファイルの例では、指定可能な各種設定を示しています。

```
apiVersion: admin.enmasse.io/v1beta1
kind: BrokeredInfraConfig
metadata:
  name: brokered-infra-config-example
spec:
  version: "0.32" (1)
  admin: (2)
    resources:
      memory: 256Mi
    podTemplate:
      metadata:
        labels:
          key: value
  broker: (3)
    resources:
      memory: 2Gi
      storage: 100Gi
    addressFullPolicy: PAGE
    globalMaxSize: 256Mb
    podTemplate: (4)
      spec:
        priorityClassName: messaging
```

1. 使用する AMQ Online バージョンを指定します。アップグレード時に、AMQ Online はこのフィールドを使用して、インフラストラクチャーを要求されたバージョンにアップグレードするかどうか
2. **admin** コンポーネントに対して設定できるオプションを指定します。
3. **broker** コンポーネントに対して設定できるオプションを指定します。**.broker.resources.storage** 設定を変更しても、既存のブローカーストレージサイズは設定されないことに注意してください。
4. **admin** コンポーネントと **broker** コンポーネントの両方で、次の **podTemplate** 要素を設定できます。

    - **metadata.labels**
    - **spec.priorityClassName**
    - **spec.tolerations**
    - **spec.affinity**
    - **spec.containers.readinessProbe**
    - **spec.containers.livenessProbe**
    - **spec.containers.resources**
    - **spec.containers.env**
      他のすべての **podTemplate** 要素は無視されます。これらの要素の詳細は、次の **Related links** セクションにある OpenShift ドキュメントを参照してください。

      readiness プローブのタイムアウトを設定する方法は ブローカインフラストラクチャー設定の readiness プローブのタイミングのオーバーライド を参照してください。

使用可能なすべての仲介インフラストラクチャー設定フィールドの詳細は、仲介インフラストラクチャー設定フィールドの表 を参照してください。

関連リンク

- **podTemplate** 設定の詳細は、次の OpenShift ドキュメントを参照してください。
    - Pod の優先度
    - テイントおよび容認 (Toleration)
    - Pod のアフィニティーおよび非アフィニティー
    - アプリケーションの正常性
    - コンピュートリソース
    - 環境変数

5.6.1.2. 仲介型インフラストラクチャー設定のプローブタイミングのオーバーライド

ブローカーリソースのプローブタイミングのデフォルト値をオーバーライドできます。たとえば、ブローカストレージが使用可能になるまでに予想以上に時間がかかる場合や、サーバーの速度が遅い場合は、デフォルト値を変更できます。

次の例は、ブローカリソースの readiness プローブの特定のデフォルト値をオーバーライドする方法を示しています。

```
apiVersion: admin.enmasse.io/v1beta1
kind: BrokeredInfraConfig
metadata:
  name: brokered-infra-config
spec:
  broker:
    ...
    podTemplate:
      spec:
        containers:
          - name: broker (1)
            readinessProbe:
              failureThreshold: 6 (2)
              initialDelaySeconds: 20 (3)
```

(1) **name** の値は、ターゲットコンテナー名と一致する必要があります。ブローカーの場合、**podTemplate** 名は **broker** です。

(2) Pod が起動してプローブが失敗したときに、Pod が readiness プローブに対して **Unready** とマークされるか、liveness プローブに対してコンテナーが再起動される前に、OpenShift が試行する回数を指定します。デフォルト値は **3** で、最小値は **1** です。

(3) コンテナーが起動してから最初のプローブが実行されるまでの秒数を指定します。

関連リンク

- プローブのタイムアウトのオーバーライドに関する OpenShift 3.11 のドキュメント
- プローブのタイムアウトのオーバーライドに関する OpenShift 4.1 ドキュメント

5.6.2. 標準インフラストラクチャー設定

StandardInfraConfig リソースは、**standard** アドレス空間によってデプロイメントされるインフラストラクチャーを設定するために使用されます。アドレス空間計画は、**spec.infraConfigRef** フィールドを使用して標準インフラストラクチャー設定を参照します。アドレス空間計画の詳細は、アドレス空間計画 を参照してください。

使用可能な標準インフラストラクチャー設定フィールドの詳細は、標準インフラストラクチャー設定フィールドの表 を参照してください。

5.6.2.1. 標準インフラストラクチャー設定例

次の標準インフラストラクチャー設定ファイルの例は、指定できるさまざまな設定を示しています。

```
apiVersion: admin.enmasse.io/v1beta1
kind: StandardInfraConfig
metadata:
  name: myconfig
spec:
  version: "0.32" (1)
```

```
  admin: 2
    resources:
      memory: 256Mi
  broker: 3
    resources:
      cpu: 0.5
      memory: 2Gi
      storage: 100Gi
    addressFullPolicy: PAGE
  router: 4
    resources:
      cpu: 1
      memory: 256Mi
    linkCapacity: 1000
    minReplicas: 1
    policy:
      maxConnections: 1000
      maxConnectionsPerHost: 1
      maxConnectionsPerUser: 10
      maxSessionsPerConnection: 10
      maxSendersPerConnection: 5
      maxReceiversPerConnection: 5
    podTemplate: 5
      spec:
        affinity:
          nodeAffinity:
            preferredDuringSchedulingIgnoredDuringExecution:
            - weight: 1
              preference:
              matchExpressions:
              - key: e2e-az-EastWest
                operator: In
                values:
                - e2e-az-East
                - e2e-az-West
```

1. 使用する AMQ Online バージョンを指定します。アップグレード時に、AMQ Online はこのフィールドを使用して、インフラストラクチャーを要求されたバージョンにアップグレードするかどうかを決定します。省略した場合、バージョンは、設定を読み取るコントローラーと同じバージョンであると想定されます。

2. **admin** コンポーネントに対して設定できるオプションを指定します。

3. **broker** コンポーネントに対して設定できるオプションを指定します。**.broker.resources.storage** 設定を変更しても、既存のブローカーストレージサイズは設定されません。

4. **router** コンポーネントに対して設定できるオプションを指定します。

5. **admin**、**broker**、および **router** コンポーネントについては、次の **podTemplate** 要素を設定できます。

    - **metadata.labels**
    - **spec.priorityClassName**
    - **spec.tolerations**

- **spec.affinity**
- **spec.containers.resources**
- **spec.containers.readinessProbe**
- **spec.containers.livenessProbe**
- **spec.containers.env**
  他のすべての **podTemplate** 要素は無視されます。これらの要素の詳細は、次の **Related links** セクションにある OpenShift ドキュメントを参照してください。

  readiness プローブのタイムアウトを設定する方法の詳細は 標準インフラストラクチャー設定の readiness プローブのタイミングのオーバーライド を参照してください。

使用可能なすべての標準インフラストラクチャー設定フィールドの詳細は、標準インフラストラクチャー設定フィールドの表 を参照してください。

関連リンク

- **podTemplate** 設定の詳細は、次の OpenShift ドキュメントを参照してください。
  - Pod の優先度
  - テイントおよび容認 (Toleration)
  - Pod のアフィニティーおよび非アフィニティー
  - アプリケーションの正常性
  - コンピュートリソース
  - 環境変数

5.6.2.2. 標準インフラストラクチャー設定のプローブタイミングのオーバーライド

ブローカーおよびルーターリソースのプローブタイミングのデフォルト値をオーバーライドできます。たとえば、ブローカストレージが使用可能になるまでに予想以上に時間がかかる場合や、サーバーの速度が遅い場合は、デフォルト値を変更できます。

次の例は、ブローカーリソースの readiness プローブタイムアウトとルーターリソースの liveness プローブの特定の既定値をオーバーライドする方法を示しています。

```
apiVersion: admin.enmasse.io/v1beta1
kind: StandardInfraConfig
metadata:
  name: standard-infra-config
spec:
  broker:
    ...
    podTemplate:
      spec:
        containers:
          - name: broker (1)
            readinessProbe:
```

```
        failureThreshold: 6 2
        initialDelaySeconds: 20 3

router:
 ...
 podTemplate:
   spec:
     containers:
       - name: router 4
         livenessProbe:
           failureThreshold: 6 5
           initialDelaySeconds: 20 6
```

1 4 **name** の値は、ターゲットコンテナー名と一致する必要があります。たとえば、ブローカー **podTemplate** の場合には、**name** は **broker** で、ルーター **podTemplate** の場合には **router** です。

2 5 Pod が起動してプローブが失敗したときに、Pod が readiness プローブに対して **Unready** とマークされるか、liveness プローブに対してコンテナーが再起動される前に、OpenShift が試行する回数を指定します。デフォルト値は **3** で、最小値は **1** です。

3 6 コンテナーが起動してから最初のプローブが実行されるまでの秒数を指定します。

関連リンク

- liveness および readiness プローブ (アプリケーションの正常性) に関する OpenShift 3.11 ドキュメント
- liveness および readiness プローブ (アプリケーションの正常性) に関する OpenShift 4.1 ドキュメント

5.7. インフラストラクチャー設定の作成と編集

新しいインフラストラクチャー設定を作成するか、既存のものを編集できます。詳しくは、インフラストラクチャー設定 を参照してください。

手順

1. サービスオペレーターとしてログインします。

```
oc login -u developer
```

2. AMQ Online がインストールされているプロジェクトに切り替えます。

```
oc project _amq-online-infra_
```

3. 既存のインフラストラクチャー設定を編集するか、次の例を使用して新しいインフラストラクチャー設定を作成します。

```
apiVersion: admin.enmasse.io/v1beta1
kind: StandardInfraConfig
metadata:
  name: myconfig
```

```
spec:
  version: "0.32"
  admin:
    resources:
      memory: 256Mi
  broker:
    resources:
      memory: 2Gi
      storage: 100Gi
    addressFullPolicy: PAGE
  router:
    resources:
      memory: 256Mi
    linkCapacity: 1000
    minReplicas: 1
```

4. 設定の変更を適用します。

```
oc apply -f standard-infra-config-example.yaml
```

5. 再起動中に Pod を監視します。

```
oc get pods -w
```

設定の変更は数分以内に適用されます。

5.8. 認証サービス

認証サービスは、メッセージングクライアントが使用できる認証エンドポイントと承認エンドポイントを設定するために使用されます。認証サービスは AMQ Online サービス Operator によって設定され、アドレス空間の作成時に指定されます。

認証サービスは、カスタムリソースとして設定されます。認証サービスには、**standard**、**external**、または **none** のいずれかのタイプがあります。

5.8.1. 標準認証サービス

standard の認証サービスタイプでは、テナント管理者は **MessagingUser** カスタムリソースを介してユーザーとそのユーザー関連のアクセス許可を管理できます。これは、Red Hat Single Sign-On インスタンスを使用してユーザー認証情報とアクセスポリシーを保存することで実現されます。一般的な使用例では、1 つの **standard** 認証サービスのみを定義する必要があります。

5.8.1.1. 標準認証サービスの例

次の例は、タイプ **standard** の認証サービスを示しています。

```
apiVersion: admin.enmasse.io/v1beta1
kind: AuthenticationService
metadata:
  name: standard
spec:
  type: standard (1)
  standard:
    credentialsSecret: (2)
```

```
    name: my-admin-credentials
  certificateSecret 3
    name: my-authservice-certificate
  resources: 4
    requests:
      memory: 2Gi
    limits:
      memory: 2Gi
  storage: 5
    type: persistent-claim
    size: 5Gi
  datasource: 6
    type: postgresql
    host: example.com
    port: 5432
    database: authdb
```

1. **type** の有効な値は、**none**、**standard**、または **external** です。
2. (オプション) シークレットには、Red Hat Single Sign-On 管理ユーザーの **admin.username** ユーザーフィールドと **admin.password** パスワードフィールドが含まれている必要があります。指定しない場合、ランダムなパスワードが生成され、シークレットに保存されます。
3. (OpenShift ではオプション) カスタム証明書を指定できます。OpenShift では、証明書が指定されていない場合、証明書が自動的に作成されます。
4. (オプション) Red Hat Single Sign-On インスタンスのリソース制限を指定できます。
5. (オプション) ストレージタイプは、**ephemeral** または **persistent-claim** として指定できます。**persistent-claim** の場合、要求のサイズも設定する必要があります。デフォルトのタイプは **ephemeral** です。
6. (オプション) Red Hat Single Sign-On が使用するデータソースを指定します。デフォルトのオプションは、埋め込み **h2** データソースです。実稼働環境で使用する場合は、**postgresql** データソースをお勧めします。

5.8.1.2. standard 認証サービスのデプロイ

standard 認証サービスを実装するには、デプロイします。

手順

1. サービス管理者としてログインします。

```
oc login -u admin
```

2. AMQ Online がインストールされているプロジェクトに切り替えます。

```
oc project amq-online-infra
```

3. **AuthenticationService** 定義を作成します。

```
apiVersion: admin.enmasse.io/v1beta1
```

```
kind: AuthenticationService
metadata:
  name: standard-authservice
spec:
  type: standard
```

4. 認証サービスをデプロイします。

```
oc create -f standard-authservice.yaml
```

5.8.1.3. 高可用性 (HA) のための standard 認証サービスのデプロイ

実稼働デプロイメントでは、OpenShift の更新中またはノード障害発生時のダウンタイムを短縮するために、認証サービスを高可用性設定にする必要があります。**standard** 認証サービスを HA モードで実装するには、バックエンドとして PostgreSQL データベースを使用してデプロイします。

前提条件

- PostgreSQL データベース。

手順

1. サービス管理者としてログインします。

```
oc login -u admin
```

2. データベース認証情報を使用してシークレットを作成します。

```
oc create secret generic db-creds -n amq-online-infra --from-literal=database-user=admin --from-literal=database-password=secure-password
```

3. **AuthenticationService** 定義を作成します。

```
apiVersion: admin.enmasse.io/v1beta1
kind: AuthenticationService
metadata:
  name: standard-authservice
spec:
  type: standard
  standard:
    replicas: 2
    datasource:
      type: postgresql
      host: database.example.com
      port: 5431
      database: auth
      credentialsSecret:
        name: db-creds
```

4. 認証サービスをデプロイします。

```
oc create -f standard-authservice.yaml -n amq-online-infra
```

5.8.2. 外部認証サービス

外部 認証サービスを使用すると、AMQP SASL ハンドシェイクを使用して認証および許可ポリシーの外部プロバイダーを設定できます。この設定を使用して、既存の ID 管理システムのブリッジを実装できます。

ユースケースによっては、複数の **外部** 認証サービスを定義する場合があります。

5.8.2.1. 外部認証サービスの例

次の例は、タイプが **external** の認証サービスを示しています。

```
apiVersion: admin.enmasse.io/v1beta1
kind: AuthenticationService
metadata:
  name: my-external-1
spec:
  type: external
  realm: myrealm 1
  external:
    host: example.com 2
    port: 5671 3
    caCertSecret: 4
      name: my-ca-cert
```

1. (オプション) **レルム** は認証要求で渡されます。指定しない場合、**namespace-addressspace** の形式の識別子がレルムとして使用されます。
2. 外部認証サーバーのホスト名。
3. 外部認証サーバーのポート番号。
4. (オプション) 認証サーバーに接続するときに信頼する CA 証明書。

外部認証サーバーは、外部認証サーバー API で説明されている API を実装する必要があります。

5.8.2.2. オーバーライドを許可する外部認証サービスの例

次の例は、メッセージングテナントによるホスト名、ポート番号、およびレルムのオーバーライドを許可する **外部** タイプの認証サービスを示しています。

```
apiVersion: admin.enmasse.io/v1beta1
kind: AuthenticationService
metadata:
  name: my-external-2
spec:
  type: external
  realm: myrealm 1
  external:
    host: example.org 2
    port: 5671 3
```

```
    caCertSecret: 4
      name: my-ca-cert
    allowOverride: true 5
```

1 (オプション) **レルム** は認証要求で渡されます。指定しない場合、**namespace-addressspace** の形式の識別子がレルムとして使用されます。

2 外部認証サーバーのホスト名。

3 外部認証サーバーのポート番号。

4 (オプション) 認証サーバーに接続するときに信頼する CA 証明書。

5 (オプション) ホスト名、ポート番号、レルム、および CA 証明書に対してアドレス空間のオーバーライドを許可するかどうかを指定します。有効な値は **true** または **false** です。指定しない場合、デフォルト値は **false** です。

外部認証サーバーは、外部認証サーバー API で説明されている API を実装する必要があります。

5.8.2.3. 外部認証サーバー API

外部認証サーバーは、AMQP SASL ハンドシェイクを実装し、クライアントの接続プロパティーを読み取り、認証および承認情報を含む予想される接続プロパティーで応答する必要があります。認証サーバーは、メッセージングエンドポイントへの新しい接続が確立されるたびに、ルーターやブローカーなどのアドレス空間コンポーネントによって照会されます。

5.8.2.3.1. 認証

要求されたクライアントの ID は、SASL ハンドシェイクの **username** から読み取ることができます。その後、実装はユーザーを認証できます。

認証済み ID は、次のキー/値を持つ **authenticated-identity** マップで返されます。この例では JSON を使用していますが、接続プロパティーで AMQP マップとして設定する必要があります。

```
{
    "authenticated-identity": {
        "sub": "myid",
        "preferred_username": "myuser"
    }
}
```

5.8.2.3.2. 承認

許可は、**ADDRESS-AUTHZ** 接続機能を使用してクライアントが要求できる機能です。これが接続で設定されている場合、サーバーは提供された機能でこの機能で応答し、認証情報を接続プロパティーに追加します。

認証情報は、アドレスを、そのアドレスで許可されている操作のリストに関連付けるマップ内に格納されます。次の接続プロパティー情報には、アドレス **myqueue** および **mytopic** のポリシーが含まれています。

```
{
    "address-authz": {
        "myqueue": [
```

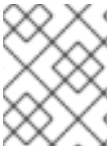
```
        "send",
        "recv"
      ],
      "mytopic": [
        "send"
      ]
    }
  }
```

許可されている操作は次のとおりです。

- **send** - ユーザーはアドレスに送信できます。
- **recv** - ユーザーはアドレスから受信できます。

5.8.3. 認証サービスなし

none 認証サービスタイプでは、任意のユーザー名とパスワードを使用するすべてのクライアントが、任意のアドレスとの間でメッセージを送受信できます。



注記

本番環境で **none** 認証サービスを使用することはお勧めしません。内部テスト環境や開発環境など、非実稼働環境でのみ使用することを意図しています。

5.8.3.1. none 認証サービスのデプロイ

none 認証サービスを実装するには、それをデプロイします。

手順

1. サービス管理者としてログインします。

```
oc login -u admin
```

2. AMQ Online がインストールされているプロジェクトに切り替えます。

```
oc project amq-online-infra
```

3. **AuthenticationService** 定義を作成します。

```
apiVersion: admin.enmasse.io/v1beta1
kind: AuthenticationService
metadata:
  name: none-authservice
spec:
  type: none
```

4. 認証サービスをデプロイします。

```
oc create -f none-authservice.yaml
```

5.9. AMQ ONLINE のロール例

AMQ Online には、直接使用したり、モデルとして使用して独自のロールを作成したりできる次のロールの例が用意されています。

サービス管理者リソースの詳細は、AMQ Online サービス管理者リソースの表 を参照してください。

メッセージングテナントリソースの詳細は、AMQ Online メッセージングテナントリソースの表 を参照してください。

表5.1 AMQ Online のロールの表の例

Role	説明
enmasse.io:tenant-view	**addresses**、**addressspaces**、**addressspaceschemas**、および **messagingusers** の **get** および **list** のアクセス許可を指定します。
enmasse.io:tenant-edit	**addresses**、**addressspaces**、**messagingusers** の **create**、**get**、**update**、**delete**、**list**、**watch**、および **patch** のアクセス許可を指定し、**addressspaceschemas** の **get** および **list** のアクセス許可を指定します。
service-admin クラスターロール	**addressplans**、**addressspaceplans**、**brokeredinfraconfigs**、**standardinfraconfigs**、および **authenticationservices** の **create**、**get**、**update**、**delete**、**list**、**watch**、および **patch** アクセス許可を指定します。

第6章 AMQ ONLINE の監視

組み込みの監視ツールをデプロイメントするか、既存の監視インフラストラクチャーを使用して、AMQ Online を監視できます。

6.1. OPENSHIFT 4 でのモニタリングの有効化

既存のモニタリングスタックを使用して OpenShift 4 で AMQ Online をモニタリングするには、ユーザーワークロードのモニタリングを 有効 にする必要があります。

6.2. (オプション) APPLICATION MONITORING OPERATOR のデプロイ

AMQ Online を監視するには、監視カスタムリソース定義に作用するオペレーターをデプロイする必要があります。そのようなオペレーターが OpenShift クラスターにインストールされている場合は、このステップをスキップできます。

手順

1. **cluster-admin** 権限を持つユーザーとしてログインしてます。

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

2. (オプション) **enmasse-monitoring** 以外の namespace にデプロイする場合は、次のコマンドを実行し、後続の手順で **enmasse-monitoring** を置き換える必要があります。

```
sed -i 's/enmasse-monitoring/my-namespace/' install/bundles/amq-online/*.yaml
```

3. enmasse-monitoring namespace を作成します。

```
oc new-project enmasse-monitoring
```

4. **monitoring-operator** リソースをデプロイします。

```
oc apply -f install/components/monitoring-operator
```

5. **monitoring-operator** コンポーネントをデプロイします。

```
oc apply -f install/components/monitoring-deployment
```

6.3. (オプション) KUBE-STATE-METRICS エージェントのデプロイ

kube-state-metrics エージェントを使用して、AMQ Online Pod を監視できます。

手順

1. **cluster-admin** 権限を持つユーザーとしてログインしてます。

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

2. **amq-online-infra** プロジェクトを選択します。

```
oc project amq-online-infra
```

3. **kube-state-metrics** コンポーネントをデプロイします。

```
oc apply -f install/components/kube-state-metrics
```

6.4. モニターリングの有効化

デフォルトのインストール設定を使用していない場合は、監視をデプロイメントする最も簡単な方法は、**enmasse-operator** デプロイメントで監視環境変数を有効にすることです。

前提条件

- Application Monitoring Operator または同じリソースを管理する Operator をインストールする必要があります。

手順

1. **amq-online-infra** namespace にラベルを付けます。

```
oc label namespace amq-online-infra monitoring-key=middleware
```

2. Operator でのモニターを有効にします。

```
oc set env deployment -n amq-online-infra enmasse-operator
ENABLE_MONITORING=true
```

6.5. アラート通知の設定

電子メールなどのアラート通知を設定するには、Alertmanager のデフォルト設定を変更する必要があります。

前提条件

- Alertmanager ドキュメント に従って、Alertmanager 設定ファイルを作成します。電子メール通知の設定ファイルの例を次に示します。

```
apiVersion: v1
kind: ConfigMap
metadata:
  labels:
    app: enmasse
  name: alertmanager-config
data:
  alertmanager.yml: |
    global:
      resolve_timeout: 5m
      smtp_smarthost: localhost
      smtp_from: alerts@localhost
      smtp_auth_username: admin
      smtp_auth_password: password
    route:
```

```
    group_by: ['alertname']
    group_wait: 60s
    group_interval: 60s
    repeat_interval: 1h
    receiver: 'sysadmins'
  receivers:
  - name: 'sysadmins'
    email_configs:
    - to: sysadmin@localhost
  inhibit_rules:
    - source_match:
        severity: 'critical'
      target_match:
        severity: 'warning'
      equal: ['alertname']
```

- Prometheus Operator が読み取れるように、Alertmanager 設定ファイルの名前を **alertmanager.yaml** にする必要があります。

手順

1. デフォルト設定を含むシークレットを削除します。

```
oc delete secret alertmanager-application-monitoring
```

2. 新しい設定を含むシークレットを作成します。

```
oc create secret generic alertmanager-application-monitoring --from-file=alertmanager.yaml
```

6.6. メトリックとルール

6.6.1. 一般的なメトリック

次のコンポーネントは、これらの一般的なメトリックをエクスポートします。

- **enmasse-operator**
- **address-space-controller**
- **standard-controller**

enmasse_version

タイプ

version

説明

バージョンラベルを使用して、AMQ Online の各コンポーネントの現在のバージョンを提供します。メトリックは常に値 **1** を返します。

例

```
enmasse_version{job="address-space-controller",version="1.0.1"} 1
enmasse_version{job="enmsse-operator",version="1.0.1"} 1
enmasse_version{job="standard-controller",version="1.0.1"} 1
```

6.6.2. アドレス空間コントローラーのメトリック

AMQ Online では、**address-space-controller** の次のメトリックを使用できます。

6.6.2.1. 概要

タイプ **enmasse_address_space_status_ready** のエクスポートされたすべてのメトリックに対して、タイプ **enmasse_address_space_status_not_ready** の対応するメトリックがあります。それぞれの値が同じになることはありません。

以下に例を示します。

```
enmasse_address_space_status_ready{name="my-address-space"} 1
enmasse_address_space_status_not_ready{name="my-address-space"} 0
```

アドレス空間の総数は、ready 状態のすべてのアドレス空間の合計と、not ready 状態のすべてのアドレス空間の合計に等しくなります。

```
enmasse_address_spaces_total == (sum(enmasse_address_space_status_ready) +
sum(enmasse_address_space_status_not_ready))
```

enmasse_address_space_status_ready

タイプ

ブール値

説明

ready 状態にある各アドレス空間を示します。

例

```
enmasse_address_space_status_ready{name="prod-space"} 1
enmasse_address_space_status_ready{name="dev-space"} 0
```

enmasse_address_space_status_not_ready

タイプ

ブール値

説明

not ready 状態にある各アドレス空間を示します。

例

```
enmasse_address_space_status_not_ready{name="prod-space"} 0
enmasse_address_space_status_not_ready{name="dev-space"} 1
```

enmasse_address_spaces_total

タイプ

ゲージ

説明

ready 状態または **not ready** 状態かに関係なく、アドレス空間の総数を返します。

例

enmasse_address_spaces_total 1

enmasse_address_space_connectors_total

タイプ

ゲージ

説明

各アドレス空間のアドレス空間コネクターの総数を返します。

例

```
enmasse_address_space_connectors_total{name="space-one"} 0
enmasse_address_space_connectors_total{name="space-two"} 2
```

6.6.3. 標準のコントローラーとエージェントのメトリック

次の **standard-controller** および **agent** メトリックは、AMQ Online の仲介アドレス空間でのみ使用できます。

6.6.3.1. 概要

アドレスの合計数は、ready 状態のアドレスの合計数と、not ready 状態のアドレスの合計数の合計に等しくなります。

```
enmasse_addresses_total == enmasse_addresses_ready_total +
enmasse_addresses_not_ready_total
```

アドレスの総数は、すべてのフェーズのアドレスの総数に等しくなります。

```
enmasse_addresses_total == enmasse_addresses_active_total +
enmasse_addresses_configuring_total + enmasse_addresses_failed_total +
enmasse_addresses_pending_total + enmasse_addresses_terminating_total
```

enmasse_addresses_total

説明

状態に関係なく、アドレス空間ごとのアドレスの総数を提供します。

タイプ

ゲージ

例

```
enmasse_addresses_total{addressspace="space-one"} 5
enmasse_addresses_total{addressspace="space-two"} 3
```

enmasse_addresses_ready_total

タイプ

ゲージ

説明

現在 ready 状態にあるアドレスの総数を提供します。

例

```
enmasse_addresses_ready_total{addressspace="space-one"} 3
enmasse_addresses_ready_total{addressspace="space-two"} 2
```

enmasse_addresses_not_ready_total

タイプ

ゲージ

説明

現在 not ready 状態のアドレスの総数を提供します。

例

```
enmasse_addresses_not_ready_total{addressspace="space-one"} 2
enmasse_addresses_not_ready_total{addressspace="space-two"} 1
```

enmasse_addresses_active_total

タイプ

ゲージ

説明

現在アクティブなフェーズにあるアドレスの総数を提供します。

例

enmasse_addresses_active_total{addressspace="space-one"} 2

enmasse_addresses_configuring_total

タイプ

ゲージ

説明

現在設定フェーズにあるアドレスの総数を提供します。

例

enmasse_addresses_configuring_total{addressspace="space-one"} 2

enmasse_addresses_failed_total

タイプ

ゲージ

説明

現在失敗フェーズにあるアドレスの総数を提供します。

例

enmasse_addresses_failed_total{addressspace="space-one"} 2

enmasse_addresses_pending_total

タイプ

ゲージ

説明

現在保留フェーズにあるアドレスの総数を提供します。

例

enmasse_addresses_pending_total{addressspace="space-one"} 2

enmasse_addresses_terminating_total

タイプ

ゲージ

説明

現在終了フェーズにあるアドレスの総数を提供します。

例

enmasse_addresses_terminating_total{addressspace="space-one"} 2

enmasse_standard_controller_loop_duration_seconds

タイプ

ゲージ

説明

最新の標準コントローラー調整ループの実行時間を秒単位で提供します。

例

enmasse_standard_controller_loop_duration_seconds 0.33

enmasse_standard_controller_router_check_failures_total

タイプ

カウンター

説明

調整ループ中のルーターチェックの失敗の合計数を示します。

例

```
enmasse_standard_controller_router_check_failures_total{addressspace="firstspace"} 0
enmasse_standard_controller_router_check_failures_total{addressspace="myspace"} 0
```

enmasse_addresses_forwarders_ready_total

タイプ

ゲージ

説明

ready 状態のアドレスフォワーダーの総数を提供します。

例

enmasse_addresses_forwarders_ready_total{addressspace="myspace"} 2

enmasse_addresses_forwarders_not_ready_total

タイプ

ゲージ

説明

not ready 状態のアドレスフォワーダーの総数を提供します。

例

enmasse_addresses_forwarders_not_ready_total{addressspace="myspace"} 0

enmasse_addresses_forwarders_total

タイプ

ゲージ

説明

準備完了状態か準備完了状態かに関係なく、アドレスフォワーダーの合計数を提供します。

例

enmasse_addresses_forwarders_total{addressspace="myspace"} 2

enmasse_address_canary_health_failures_total

タイプ

ゲージ

説明

プローブアドレスへのメッセージの送受信に失敗したために失敗したヘルスチェックの総数。

例

enmasse_address_canary_health_failures_total{addressspace="myspace"} 2

enmasse_address_canary_health_check_failures_total

タイプ

ゲージ

説明

コントローラーエラーが原因で失敗したヘルスチェック実行の試行の合計数。

例

enmasse_address_canary_health_check_failures_total{addressspace="myspace"} 1

6.6.4. ルール

このセクションでは、AMQ Online で PrometheusRule CRD を使用してインストールされた Prometheus ルールについて詳しく説明します。AMQ Online では、2 種類の Prometheus ルールを使用できます。

- 記録: 時系列の新しいセットとして保存された、事前に計算された式。
- 警告: **true** と評価されたときにアラートをトリガーする式。

6.6.4.1. Records

レコードは、時系列の新しいセットとして保存される事前計算された式である Prometheus ルールの一種です。AMQ Online では、次のレコードを利用できます。

enmasse_address_spaces_ready_total

説明

enmasse_address_space_status_ready を、**ready** 状態のアドレスの総数を提供する単一のゲージタイプメトリックに集約します。

式

```
sum by(service, exported_namespace) (enmasse_address_space_status_ready)
```

例

```
enmasse_address_spaces_ready_total{exported_namespace="prod_namespace",service="address-space-controller"} 1
```

enmasse_address_spaces_not_ready_total

説明

enmasse_address_space_not_status_ready を、**not ready** 状態のアドレスの総数を提供する単一のゲージタイプメトリックに集約します。

式

```
sum by(service, exported_namespace) (enmasse_address_space_status_not_ready)
```

例

```
enmasse_address_spaces_not_ready_total{exported_namespace="prod_namespace",service="address-space-controller"} 1
```

enmasse_component_health

説明

各 **address-space-controller** と **api-server** が稼働しているかどうかを示すブール型のメトリックを提供します。

式

```
up{job="address-space-controller"} or on(namespace) (1 - absent(up{job="address-space-controller"}))
up{job="api-server"} or on(namespace) (1 - absent(up{job="api-server"}))
```

例

```
enmasse_component_health{job="address-space-controller"} 1
enmasse_component_health{job="api-server"} 1
```

6.6.4.2. アラート

アラートは、true と評価されたときにアラートをトリガーする式である Prometheus ルールの一種です。AMQ Online では、次のアラートを利用できます。

ComponentHealth

説明

コンポーネントが正常な状態でない場合にトリガーされます。

式

component_health == 0

AddressSpaceHealth

説明

1つ以上のアドレス空間が **ready** 状態にない場合にトリガーされます。

式

enmasse_address_spaces_not_ready_total > 0

AddressHealth

説明

1つ以上のアドレスが **ready** 状態にない場合にトリガーされます。

式

enmasse_addresses_not_ready_total > 0

6.7. テナントメトリックの有効化

ブローカーおよびルーターからのメトリックは、システム管理メトリックを公開せずにテナントに公開できます。テナントメトリクスを公開するには、**amq-online-infra 以外** の namespace (理想的には関連するアドレス空間の namespace) にサービスモニターを作成します。

前提条件

- Prometheus Operator によって提供される **servicemonitor** Custom Resource Definition をインストールする必要があります。
- テナントには、独自の監視スタックがインストールされている必要があります。

手順

- namespace セレクターとして、**monitoring-key: enmasse-tenants** および **amq-online-infra** のラベルと一致するように設定されたセレクターを使用して、**servicemonitor** リソースを作成します。サービスモニターの例を以下に示します。

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: enmasse-tenants
  labels:
    app: enmasse
spec:
  selector:
    matchLabels:
      monitoring-key: enmasse-tenants
  endpoints:
  - port: health
```

```
  namespaceSelector:
    matchNames:
      - amq-online-infra
```

- テナントのモニタリングスタックに、サービスモニターの namespace のサービスモニターの読み取りアクセス許可があることを確認しますが、これにより service-admin メトリックも公開されるため、**amq-online-infra** 内にはアクセス許可がありません。

6.8. QDSTAT の使用

qdstat を使用して、AMQ Online サービスを監視できます。

6.8.1. qdstat を使用したルーター接続の表示

qdstat を使用してルーター接続を表示できます。

手順

1. コマンドラインで次のコマンドを実行して、次の手順で必要な **podname** の値を取得します。

```
oc get pods
```

2. コマンドラインで、次のコマンドを実行します。

```
oc exec -n namespace -it qdrouterd-podname -- qdstat -b 127.0.0.1:7777 -c

Connections
  id   host              container                       role    dir  security
authentication              tenant

========================================================================
========================================================================
===
  3    172.17.0.9:34998     admin-78794c68c8-9jdd6              normal  in   TLSv1.2(ECDHE-
RSA-AES128-GCM-SHA256)  CN=admin,O=io.enmasse(x.509)
  12   172.30.188.174:5671  27803a14-42d2-6148-9491-a6c1e69e875a  normal  out
TLSv1.2(ECDHE-RSA-AES128-GCM-SHA256)  x.509
  567  127.0.0.1:43546      b240c652-82df-48dd-b54e-3b8bbaef16c6  normal  in   no-security
PLAIN
```

6.8.2. qdstat を使用したルーターアドレスの表示

qdstat を使用してルーターアドレスを表示できます。

手順

1. コマンドラインで次のコマンドを実行して、次の手順で必要な **podname** の値を取得します。

```
oc get pods
```

2. 以下のコマンドを実行します。

```
oc exec -n namespace -it qdrouterd-podname -- qdstat -b 127.0.0.1:7777 -a
```

```
Router Addresses
  class   addr                phs  distrib      in-proc  local  remote  cntnr  in    out   thru  to-proc
from-proc

=================================================================================================
  local   $_management_internal      closest      1        0      0       0      0     0     0     588
588
  link-in   $lwt                   linkBalanced  0        0      0       0      0     0     0     0
  link-out  $lwt                   linkBalanced  0        0      0       0      0     0     0     0
  mobile    $management          0   closest      1        0      0       0      601   0     0     601   0
  local     $management              closest      1        0      0       0      2,925 0     0     2,925 0
  local     qdhello                  flood        1        0      0       0      0     0     0     0     5,856
  local     qdrouter                 flood        1        0      0       0      0     0     0     0     0
  topo      qdrouter                 flood        1        0      0       0      0     0     0     0     196
  local     qdrouter.ma              multicast    1        0      0       0      0     0     0     0     0
  topo      qdrouter.ma              multicast    1        0      0       0      0     0     0     0     0
  local     temp.VTXOKyyWsq7OEei     balanced     0        1      0       0      0     0     0     0
0
  local     temp.k2RGQNPe6sDMvz4     balanced     0        1      0       0      0     3,511 0     0
3,511
  local     temp.xg+y8l_Tr4Y94LA     balanced     0        1      0       0      0     5     0     0
5
```

6.8.3. qdstat を使用したルーターリンクの表示

qdstat を使用して、ルーターリンクを表示できます。

手順

1. コマンドラインで次のコマンドを実行して、次の手順で必要な **podname** の値を取得します。

```
oc get pods
```

2. コマンドラインで、次のコマンドを実行します。

```
oc exec -n namespace -it qdrouterd-podname -- qdstat -b 127.0.0.1:7777 -l

Router Links
  type      dir  conn id  id  peer  class   addr                phs  cap  undel  unsett  del   presett
psdrop  acc   rej  rel  mod  admin    oper

=================================================================================================
=================================================================================================

  endpoint  in   3       8                                  250  0      0      3829  0       0      3829  0
0    0    enabled  up
  endpoint  out  3       9        local   temp.k2RGQNPe6sDMvz4      250  0      0      3829  3829
0       0    0    0    0    enabled  up
  endpoint  in   12      10                                 250  0      0      5     0       0      5     0    0
0    enabled  up
  endpoint  out  12      11       local   temp.xg+y8l_Tr4Y94LA      250  0      0      5     5       0
0    0    0    0    enabled  up
  endpoint  in   645     26       mobile  $management          0    50   0      0      1     0       0
```

```
1    0   0   0   enabled  up
  endpoint  out  645     27       local   temp.0BrHJ1O+fi6whyg      50  0     0      0    0       0
0    0   0   0   enabled  up
```

6.8.4. qdstat を使用したリンクルートの表示

qdstat を使用してリンクルートを表示できます。

手順

1. コマンドラインで次のコマンドを実行して、次の手順で必要な **podname** の値を取得します。

```
oc get pods
```

2. コマンドラインで、次のコマンドを実行します。

```
oc exec -n namespace -it qdrouterd-podname -- qdstat -b 127.0.0.1:7777 --linkroutes

Link Routes
  address  dir  distrib       status
  ====================================
  $lwt     in   linkBalanced  inactive
  $lwt     out  linkBalanced  inactive
```

第7章 AMQ ONLINE の操作手順

7.1. コンポーネントを再起動してセキュリティー修正を取得する

CVE のイメージ更新を取得するには、AMQ Online コンポーネントを再起動する必要があります。スクリプトは、**script** フォルダー内の AMQ Online インストールファイルで提供されます。すべてのコンポーネントを再起動するには、すべてのスクリプトを実行します。

7.1.1. Operator の再起動

メッセージングシステムに影響を与えることなく、Operator を再起動できます。

手順

- **restart-operators.sh** スクリプトを実行します。

```
./scripts/restart-operators.sh amq-online-infra
```

7.1.2. 認証サービスの再起動

認証サービスの再起動は、新しいメッセージング接続に一時的に影響します。認証サービスが再起動されても、既存の接続は引き続き機能します。

手順

- **restart-authservices.sh** スクリプトを実行します。

```
./scripts/restart-authservices.sh amq-online-infra
```

7.1.3. ルーターの再起動

メッセージングルーターは、**standard** のアドレス空間タイプでのみデプロイメントされます。このスクリプトは、ルーターの少なくとも 2 つのレプリカが実行していることを前提としており、ローリング再起動を実行します。再起動中のルーターに接続されているメッセージングクライアントは切断され、別のルーターでサービスを受けるには再接続する必要があります。

手順

- **restart-routers.sh** スクリプトを実行します。これには、少なくとも 1 つのルーターが使用可能である必要があります。

```
./scripts/restart-routers.sh amq-online-infra 1
```

7.1.4. ブローカーの再始動

brokered アドレス空間タイプの場合は、ブローカーを再起動すると、ブローカーの再起動中にメッセージングクライアントに一時的なダウンタイムが発生します。**standard** アドレス空間タイプの場合は、メッセージングクライアントがメッセージングルーターから切断されませんが、再起動するブローカーに格納されているメッセージをクライアントが消費することはできません。

手順

- **restart-brokers.sh** スクリプトを実行します。

```
./scripts/restart-brokers.sh amq-online-infra
```

7.2. ルーターログの表示

standard アドレス空間タイプの場合、ルーターログを表示して、クライアントが接続しない問題やメッセージの送受信に関する問題をトラブルシューティングできます。

手順

1. すべてのルーター Pod を一覧表示し、関連するアドレス空間の Pod を選択します。

```
oc get pods -l name=qdrouterd -o go-template --template '{{range .items}}{{.metadata.name}}
{{"\t"}}{{.metadata.annotations.addressSpace}}{{"\n"}}{{end}}'
```

2. Pod のログを表示します。

```
oc logs pod -c router
```

7.3. ブローカーログの表示

brokered または **standard** アドレス空間タイプの場合は、ブローカーログを表示して、クライアントが接続しない問題やメッセージの送受信に関する問題をトラブルシューティングできます。

手順

1. すべてのブローカー Pod を一覧表示し、関連するアドレス空間の Pod を選択します。

```
oc get pods -l role=broker -o go-template --template '{{range .items}}{{.metadata.name}}
{{"\t"}}{{.metadata.annotations.addressSpace}}{{"\n"}}{{end}}'
```

2. Pod のログを表示します。

```
oc logs pod
```

7.4. ルーターの AMQP プロトコルトレースを有効にする

診断目的で、ルーターの AMQP プロトコルトレースを有効にできます。これは、クライアント接続またはメッセージの送受信に関する問題のトラブルシューティングに役立ちます。ルーターのプロトコルトレースを有効にする方法は 2 つあります。

- **qdmange** コマンドを使用して、単一ルーターのプロトコルトレースを動的に有効/無効にすることができます。この方法により、ルーターを再起動する必要がなくなります。次回ルーターを再起動すると、設定は失われます。

- または、その **standardinfraconfig** を使用してすべてのアドレス空間のすべてのルーターのプロトコルトレースを有効にする **standardinfraconfig** に設定を適用できます。この方法により、すべてのルーターが再起動します。

警告

プロトコルトレースを有効にすると、ルーターの CPU オーバーヘッドが増加し、メッセージングパフォーマンスが低下する可能性があります。また、ログ保持システムに関連するディスク容量の要件が増加する可能性があります。したがって、プロトコルトレースを有効にする時間をできるだけ短くすることをお勧めします。

7.4.1. 単一ルーターのプロトコルトレースを動的に有効にする

手順

1. サービスオペレーターとしてログインします。

```
oc login -u developer
```

2. AMQ Online がインストールされているプロジェクトに切り替えます。

```
oc project amq-online-infra
```

3. すべてのルーター Pod を一覧表示し、関連するアドレス空間の Pod を選択します。

```
oc get pods -l name=qdrouterd -o go-template --template '{{range .items}}{{.metadata.name}}
{{"\t"}}{{.metadata.annotations.addressSpace}}{{"\n"}}{{end}}'
```

4. 単一のルーターのプロトコルトレースを有効にします。

```
echo '{"enable":"trace+"}' | oc exec qdrouterd-podname --stdin=true --tty=false -- qdmanage
update -b 127.0.0.1:7777 --type=log --name=log/PROTOCOL --stdin
```

5. プロトコルトレースを含む Pod のログを表示します。

```
oc logs pod
```

6. プロトコルトレースを無効にします。

```
echo '{"enable":"info"}' | oc exec qdrouterd-podname --stdin=true --tty=false -- qdmanage
update -b 127.0.0.1:7777 --type=log --name=log/PROTOCOL --stdin
```

7.4.2. StandardInfraConfig 環境変数を使用してプロトコルトレースを有効にする

手順

1. サービスオペレーターとしてログインします。

```
oc login -u developer
```

2. AMQ Online がインストールされているプロジェクトに切り替えます。

```
oc project amq-online-infra
```

3. 関連するアドレス空間の **addresspaceplan** 名を決定します。

```
oc get addressspace -n namespace address-space-name --output 'jsonpath={.spec.plan}
{"\n"}'
```

4. **addressspaceplan** 名の **standardinfraconfig** 名を決定します。

```
oc get addressspaceplan address-space-plan --output 'jsonpath={.spec.infraConfigRef}
{"\n"}'
```

5. その **standardinfraconfig** を使用して、すべてのアドレス空間のすべてのルーターのプロトコルトレースを有効にします。

```
oc patch standardinfraconfig standardinfraconfig-name --type=merge -p '{"spec":{"router":
{"podTemplate":{"spec":{"containers":[{"env":
[{"name":"PN_TRACE_FRM","value":"true"}],"name":"router"}]}}}}}'
```

6. プロトコルトレースを含む Pod のログを表示します。

```
oc logs pod
```

7. プロトコルトレースを無効にします。

```
oc patch standardinfraconfig standardinfraconfig-name --type=merge -p '{"spec":{"router":
{"podTemplate":{"spec":{"containers":[{"env":
[{"name":"PN_TRACE_FRM"}],"name":"router"}]}}}}}'
```

7.5. ブローカーの AMQP プロトコルトレースを有効にする

診断目的で、ブローカーの AMQP プロトコルトレースを有効にすることができます。これは、メッセージの送受信に関する問題のトラブルシューティングに役立ちます。

プロトコルトレースを有効にするには、**standardinfraconfig** (標準アドレス空間の場合) または **brokeredinfraconfig** (仲介アドレス空間の場合) に設定を適用し、その設定を使用してすべてのアドレス空間の全ブローカーのプロトコルトレースを有効にします。この設定を適用すると、ブローカーが再起動します。

警告

プロトコルトレースを有効にすると、ブローカーの CPU オーバーヘッドが増加し、メッセージングパフォーマンスが低下する可能性があります。また、ログ保持システムに関連するディスク容量の要件が増加する可能性があります。したがって、プロトコルトレースを有効にする時間をできるだけ短くすることをお勧めします。

手順

1. サービスオペレーターとしてログインします。

```
oc login -u developer
```

2. AMQ Online がインストールされているプロジェクトに切り替えます。

```
oc project amq-online-infra
```

3. 関連するアドレス空間の **addresspaceplan** 名を決定します。

```
oc get addressspace -n namespace address-space-name --output 'jsonpath={.spec.plan}
{"\n"}'
```

4. **addressspaceplan** 名の **standardinfraconfig** または **brokeredinfraconfig** 名を決定します。

```
oc get addressspaceplan address-space-plan --output 'jsonpath={.spec.infraConfigRef}
{"\n"}'
```

5. その **standardinfraconfig** または **brokeredinfraconfig** を使用して、すべてのアドレス空間のすべてのブローカーのプロトコルトレースを有効にします。

```
oc patch infraconfig-resource infraconfig-name --type=merge -p '{"spec":{"broker":
{"podTemplate":{"spec":{"containers":[{"env":
[{"name":"PN_TRACE_FRM","value":"true"}],"name":"broker"}]}}}}}'
```

6. プロトコルトレースを含む Pod のログを表示します。

```
oc logs pod
```

7. プロトコルトレースを無効にします。

```
oc patch infraconfig-resource infraconfig-name --type=merge -p '{"spec":{"broker":
{"podTemplate":{"spec":{"containers":[{"env":
[{"name":"PN_TRACE_FRM"}],"name":"broker"}]}}}}}'
```

7.6. AMQ BROKER 管理インターフェイスを使用してブローカーの状態を調べる

アドレス空間に関連付けられた Broker に問題があると疑われる場合は、組み込みの 管理インターフェイス を使用して直接 Broker の状態を調べることができます。AMQ Online は、AMQ Broker の CLI と JMX (Jolokia 経由) を公開します。AMQ Broker Console は公開されません。

手順

1. サービス管理者としてログインします。

```
oc login -u admin
```

2. AMQ Online がインストールされているプロジェクトに切り替えます。

```
oc project amq-online-infra
```

3. アドレス空間の uuid を取得します。

```
oc get addressspace myspace -o jsonpath='{.metadata.annotations.enmasse\.io/infra-uuid}'
```

4. アドレス空間のブローカーサポート認証情報 (ユーザー名とパスワード) を取得します。

```
oc get secret broker-support-uuid  --template='{{.data.username}}' | base64 --decode
oc get secret broker-support-uuid  --template='{{.data.password}}' | base64 --decode
```

5. ブローカー Pod 名を特定します。

```
oc get pods -l infraUuid=uuid,role=broker
```

標準アドレスでは、多くのブローカーが存在する場合があります。特定のキューをホストしているブローカーを特定するには、次のコマンドを使用します。

```
oc get address address-resource-name  -o jsonpath="
{.status.brokerStatuses[*].containerId}"
```

6. ブローカーの Pod でサポートコマンドを実行します。
AMQ Broker CLI コマンドを実行するには、次のようなコマンドを使用します。

```
oc exec broker-pod-name -- /opt/amq/bin/artemis address show --user username --
password password
```

AMQ Broker Jolokia JMX コマンドを実行するには、次のようなコマンドを使用します。

```
oc exec broker-pod-name -- curl --silent --insecure --user username:_password_ -H
"Origin: https://localhost:8161"
'https://localhost:8161/console/jolokia/read/org.apache.activemq.artemis:broker="broker pod
name"/AddressMemoryUsage'
```

重要

URL 内のブローカー Pod 名を二重引用符で囲む必要があります。上記のコマンドに示すように、URL 全体を一重引用符で囲んで、コマンドシェルからそれらを保護してください。それらが存在しない場合は、認証エラーが発生します。

第8章 AMQ ONLINE 設定のサイジングガイドライン

次の情報は、AMQ Online インストールのサイズを決定する方法に関するガイドラインです。より具体的には、これらのガイドラインは、ユースケースに基づいたコンポーネントとプランの具体的な設定の推奨事項と、設定設定を調整する際のトレードオフを説明します。AMQ Online のサイジングには、次の設定が含まれます。

- ブローカー
- ルーター (標準アドレス空間のみ)
- Operator
- プラン

たとえば、各アドレス空間タイプには、アドレス計画を作成する際に考慮する必要がある特定の特徴があります。

アドレス空間のタイプとそのセマンティクスの詳細は、アドレス空間 を参照してください。

注記

AMQ Online コンポーネントを適切にサイジングするには、OpenShift クラスターに関する次の点も考慮する必要があります。

- OpenShift クラスターには、要求されたリソースを処理するのに十分な容量が必要です。OpenShift ノードが 4 GB のメモリーで設定されている場合は、ブローカーとルーターを 4 GB を超えるメモリーサイズで設定することはできません。
- 各アドレス空間は専用のインフラストラクチャーを作成するため、クラスアスターの容量が、アドレス空間の数が増得た場合のニーズにも対応できるようにする必要があります。
- アフィニティーと許容範囲を使用すると、メッセージングインフラストラクチャーで使用できるノードが制限される場合もあります。

8.1. ブローカーコンポーネントのサイジング

ブローカーは、アドレス空間のタイプに応じて **BrokeredInfraConfig** および **StandardInfraConfig** リソースを使用して設定されます。ブローカーのサイズを決定するときは、次の点を考慮してください。

- 平均メッセージサイズ
- 保存されたメッセージの数
- キューとトピックの数
- アドレスのフルポリシー

注記

AMQ Online では、ブローカーに割り当てられるメモリーの合計量のみを制限できます。個々のアドレスが使用するメモリー量を制限することはできません。

ブローカーはすべてのメッセージをディスクに永続化します。**BLOCK**、**FAIL**、または **DROP** アドレス

のフルポリシーが指定されている場合、永続化できるメッセージの数は、ブローカーのメモリー量に制限されます。**PAGE** アドレスのフルポリシーを使用すると、メモリーに保持できるよりも多くのメッセージを格納できますが、ディスクからのデータの読み取りによるパフォーマンスの低下が生じる可能性があります。したがって、ページングは、システム内のメッセージが大きい場合やメッセージのバックログが大きい場合に役立ちます。

8.1.1. ブローカーコンポーネント設定の使用例

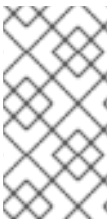
キューごとに最大 1000 個のメッセージが保存され、平均メッセージサイズが 128 KB の 10 個のキューがある場合は、メッセージを保存するために必要なストレージ容量は次のとおりです。

```
10 queues * 1000 messages * (128 + (128 kB * 1024)) = 1.25 GB
```

さらに、ブローカーには約 50 MB の固定ストレージフットプリントがあります。

ブローカに必要なメモリー量は、どのアドレスフルポリシーが指定されているかによって異なります。**PAGE** ポリシーを使用すると、メッセージがジャーナル (常にメモリーに収まる必要がある) とは別に保存されるため、メモリー要件を減らすことができます。**FAIL**、**BLOCK**、または **DROP** ポリシーが指定されている場合は、メッセージが永続化されている場合でも、すべてのメッセージをメモリーに保持する必要があります。

また、ブローカーと JVM の実行に関連する一定のメモリーコストもあります。メッセージの保存に使用できるメモリーは、ブローカ設定で設定されたメモリーから自動的に導出され、JVM メモリーの半分に設定され、さらにシステムメモリーの半分に設定されます。



注記

standard のアドレス空間タイプでは、複数のブローカーインスタンスが作成される場合があります。これらのブローカーインスタンスのサイジングは、アドレスプランの設定と、別のブローカーが生成される前に各ブローカーが処理できると予想されるアドレスの数にも依存します。

8.1.1.1. ページングを使用しないブローカーコンポーネントの設定例

PAGE ポリシーを使用しないブローカー設定の場合は、アドレスごとに追加の 5% のブックキーピングオーバーヘッドを考慮する必要があります (**1.05 * 1.25 = 1.35 GB**)。

```
apiVersion: admin.enmasse.io/v1beta1
kind: BrokeredInfraConfig
metadata:
  name: cfg1
spec:
  broker:
    addressFullPolicy: FAIL
    globalMaxSize: 1.35Gb
    resources:
      memory: 8Gi
      storage: 2Gi
  ...
```

8.1.1.2. ページングを使用したブローカーコンポーネントの設定例

ページングが有効になっている場合は、元の式を変更して、メッセージへの参照のみを考慮し、1000 個の処理中のメッセージをメモリーに保持することができます。

```
(1000 messages * 1000 * 128 kB) + (10 queues * 128 kB * 1024) = 123.5 MB
```

したがって、次の設定例に示すように、ブローカーに指定されたメモリーの量を減らすことができるようになりました。

```
apiVersion: admin.enmasse.io/v1beta1
kind: BrokeredInfraConfig
metadata:
  name: cfg1
spec:
  broker:
    addressFullPolicy: PAGE
    globalMaxSize: 124Mb
    resources:
      memory: 1Gi
      storage: 2Gi
  ...
```

8.1.2. ブローカーのスケーリング (標準アドレス空間のみ)

ブローカーはオンデマンドでデプロイされます。つまり、タイプ **queue** または **topic** のアドレスが作成されたときにデプロイされます。デプロイされるブローカーの数は、**AddressSpacePlan** 設定で指定されたリソース制限によって制限されます。次の **AddressSpacePlan** 設定例では、アドレス空間ごとに合計で 4 つのブローカーの制限を指定しています。

```
apiVersion: admin.enmasse.io/v1beta2
kind: AddressSpacePlan
metadata:
  name: cfg1
spec:
  resourceLimits:
    broker: 4.0
  ...
```

容量に関しては、ブローカーのメモリー要件に制限を掛けます。

ブローカーインスタンスの数は、1 と、さまざまなアドレスに使用される **AddressPlan** に基づいて指定された最大制限の間で動的にスケーリングされます。**AddressPlan** は、アドレスに必要なブローカーの割合を指定します。プランで指定された分数に、このプランを参照するアドレスの数を掛けてから、切り上げて、必要なブローカーレプリカの数を生成します。

AddressPlan の 設定例

```
apiVersion: admin.enmasse.io/v1beta2
kind: AddressPlan
metadata:
  name: plan1
spec:
  ...
  resources:
    broker: 0.01
```

アドレスプランとして **plan1** を使用して 110 個のアドレスを作成する場合、ブローカーのレプリカの数は **ceil(110 アドレス * 0.01 ブローカー) = 2 レプリカ** です。

ブローカーの総数は、アドレス空間プランのリソース制限によって制限されます。

8.2. ルーターコンポーネントのサイジング

ルーターは、**StandardInfraConfig** リソースで設定されます。ルーターのサイジングを決定する際は、次のことを考慮してください。

- アドレス数
- 接続数とリンク数
- リンク容量

ルーターは状態を保持しないため、永続ストレージは必要ありません。

アドレス設定自体には、大量のルーターメモリーは必要ありません。ただし、キューとサブスクリプションには、アドレスごとにルーターとブローカーの間に追加の 2 つのリンクが必要です。

リンクの総数は、キュー/サブスクリプションの数とクライアントリンクの数の 2 倍になります。各リンクには、そのリンクのルーティングメッセージを処理するために、ルーター内のメタデータとバッファーが必要です。

ルーターのリンク容量は、ルーターがリンクごとに処理できるメッセージの数に影響します。リンク容量をより高い値に設定すると、パフォーマンスが向上する可能性がありますが、送信者がリンクをいっぱいにしている場合に、処理中のメッセージを保持するために使用されるメモリーが増える可能性があります。多数の接続とリンクがある場合は、メモリー使用量のバランスをとるために、より低い値を指定することを検討してください。

さらに、ルーターはメッセージヘッダーを解析し、メッセージの処理と解決を管理し、その他のリンクごとのアクティビティーを実行する必要があります。リンクごとのコストは、リンク容量とメッセージサイズの定数係数を使用して導き出すことができます。この要因は、メッセージのサイズによって異なります。次の表は、さまざまなメッセージサイズ範囲に対するこの係数の概算を示しています。

表8.1 リンク倍率

メッセージサイズ (バイト)	ファクター
20-1000	18,000
1000-4000	22,000
4000-10,000	30,000
>10,000	50,000

8.2.1. ルーターコンポーネントのサイジングの使用例

次の使用例を見てみましょう。

- 500 のエニーキャストと 1000 のキューに入れられたアドレス
- 10,000 の接続クライアント (クライアントごとに 1 つのリンク)
- リンク容量 10

- 512 バイトの平均メッセージサイズ

測定に基づくと、エニーキャストアドレスごとに推定 7 kB のオーバーヘッドが現実的であるため、次のようになります。

```
500 anycast addresses * 7 kB overhead per address = 3.5 MB
```

キューとトピックのメモリー使用量は、エニーキャストアドレスのメモリー使用量よりもわずかに高く、アドレスごとに 32 kB のオーバーヘッドが推定されます。さらに、各ルーター/ブローカーリンクは、最大で **linkCapacity** メッセージ配信を追跡することができます。また、最悪のシナリオを考慮して、リンク容量に乗数を掛ける必要があります。

```
(1000 queued addresses * 32,768) + (2000 * 18,000 link multiplication factor * 100 links) = 374 MB
```

クライアント接続/リンクのメモリー使用量:

```
10,000 clients * 10 link capacity * 18,000 link multiplication factor = 1717 MB
```

注記

クライアント接続/リンクのメモリー使用量は、ルーターインスタンスの数で割ることができます。

N 台のルーターがある場合、この設定に必要なルーターメモリーの総量は、50 MB の固定ベースメモリーを含めて、**50 + 3.5 + (374 + 1717)/N MB** です。

接続とリンクの最大数を超えないようにするために、ルーターポリシーも適用できます。次の設定例は、ルーターポリシーが指定された 2 つのルーターを示しています。

```
apiVersion: admin.enmasse.io/v1beta1
kind: StandardInfraConfig
metadata:
  name: cfg1
spec:
  router:
    resources:
      memory: 1100Mi
    linkCapacity: 10
    policy:
      maxConnections: 5000
      maxSessionsPerConnection: 1
      maxSendersPerConnection: 1
      maxReceiversPerConnection: 1
  ...
```

8.2.2. 高可用性 (HA)

高可用性 (HA) 用にルーターを設定するには、必要なルーターレプリカの最小数にルーターあたりのメモリー量を掛けて、予想されるメモリー使用量を計算します。すべての接続とリンクはすべてのルーターに分散されますが、1 つのルーターに障害が発生した場合は、それらの接続とリンクが残りのルーターに再分散されるように計画する必要があります。

8.2.3. ルーターのスケーリング

ルーターは、**StandardInfraConfig** リソースの **minReplicas** と **AddressSpacePlan** で指定された **resourceLimits.router** に指定された制限内で、オンデマンドで動的にスケーリングされます。ルーターの数を最大 4 つに制限し、HA の目的で最低 2 つのルーターが必要な場合は、次の設定が必要です。

```
apiVersion: admin.enmasse.io/v1beta1
kind: StandardInfraConfig
metadata:
  name: cfg1
spec:
  router:
    minReplicas: 2
  ...
---
apiVersion: admin.enmasse.io/v1beta2
kind: AddressSpacePlan
metadata:
  name: plan1
spec:
  infraConfigRef: cfg1
  resourceLimits:
    router: 4
  ...
```

容量に関しては、ルーターのメモリー要件にリソース制限を掛けます。その後、ルーターは、アドレス空間の **AddressSpacePlan** で指定されたリソース制限までスケールアップします。

ルーターレプリカの数は、さまざまなアドレスに使用される **AddressPlan** に基づいて、最小制限と最大制限の間で動的にスケーリングされます。**AddressPlan** は、アドレスに必要なルーターの割合を表します。計画で定義された割合に、この計画を参照するアドレスの数を掛けてから、切り上げて目的のルーターレプリカの数を生成します。

AddressPlan の設定例:

```
apiVersion: admin.enmasse.io/v1beta2
kind: AddressPlan
metadata:
  name: plan1
spec:
  ...
  resources:
    router: 0.01
```

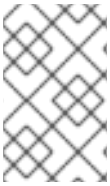
アドレスプランとして **plan1** を使用して 110 個のアドレスを作成する場合、ルーターのレプリカの数は **ceil (110 個のアドレス * 0.01 ルーター) = 2 個のレプリカ** です。

レプリカの数がアドレス空間計画の制限を超えると、最大数を超えるアドレスは **Pending** 状態のままになり、問題を説明するエラーメッセージが **Address** ステータスセクションに表示されます。

8.3. OPERATOR コンポーネントのサイジング

Operator コンポーネントは、すべてのアドレス設定を読み取り、これらの設定をルーターとブローカーに適用するロールを担っています。アドレスの数に比例して演算子コンポーネントのサイズを設定することが重要です。

standard アドレス空間では、**admin** Pod に **agent** と **standard-controller** の 2 つのプロセスが含まれています。これらのプロセスを個別にサイズ設定することはできませんが、両方のメモリー使用量はアドレス数に比例します。**brokered** アドレス空間には、**agent** プロセスが 1 つだけ存在します。



注記

オペレータープロセスは、JVM または Node.JS 仮想マシンのいずれかで実行されています。これらのプロセスのメモリー量は、アドレス設定自体に必要なメモリー量の 2 倍にすることをお勧めします。

8.3.1. Operator コンポーネントの設定例

各アドレスは、Operator プロセスに約 20 kB のオーバーヘッドを追加します。1500 個のアドレスがある場合、Operator プロセスにはさらに **1500 * 2 kB = 30 MB** が必要です。

さらに、これらのプロセスには 256 MB の基本メモリー要件があります。したがって、必要なオペレータメモリーの合計は **256 MB + 30 MB = 286 MB** です。この値は、**StandardInfraConfig** リソースと **BrokeredInfraConfig** リソースの両方で設定できます。

```
apiVersion: admin.enmasse.io/v1beta1
kind: StandardInfraConfig
metadata:
  name: cfg1
spec:
  admin:
    resources:
      memory: 300Mi
  ...
```

8.4. プランのサイズ

プランは、ブローカーとルーターのサイジングセクションに示されているように、**standard** アドレス空間での動的スケーリングを有効にします。クラスターレベルでは、プランとインフラストラクチャー構成設定の組み合わせによって、クラスターにデプロイできる Pod の最大数が決まります。AMQ Online は、作成できるアドレス空間の数の制限をサポートしていないため、ポリシーを適用して、アドレス空間の作成を許可するユーザーを制限することをお勧めします。このようなポリシー設定は、標準の OpenShift ポリシーを介して処理できます。

容量計画の観点から、Pod の最大数と、特定のアドレス空間で消費できるメモリーの最大量を計算すると便利です。スクリプトを使用してこの計算を行うには 、check-memory 計算スクリプトの実行 を参照してください。

8.4.1. チェックメモリー計算スクリプトの実行

このスクリプトを使用して、Pod の最大数と、特定のアドレス空間で消費できるメモリーの最大量を計算できます。

このスクリプトでは、メモリーは **Mi** 単位、ストレージは **Gi** 単位で指定されているものとします。また、スクリプトが意図したとおりに機能するには、**admin**、**router**、および **broker** の 3 つのコンポーネントにすべて制限を指定する必要があります。

手順

1. 次のスクリプトを **check-memory.sh** として保存します。

```
#!/usr/bin/env bash
PLAN=$1

total_pods=0
total_memory_mb=0
total_storage_gb=0

routers=$(oc get addressspaceplan $PLAN -o jsonpath='{.spec.resourceLimits.router}')
brokers=$(oc get addressspaceplan $PLAN -o jsonpath='{.spec.resourceLimits.broker}')
infra=$(oc get addressspaceplan $PLAN -o jsonpath='{.spec.infraConfigRef}')

operator_memory=$(oc get standardinfraconfig $infra -o
jsonpath='{.spec.admin.resources.memory}')
broker_memory=$(oc get standardinfraconfig $infra -o
jsonpath='{.spec.broker.resources.memory}')
broker_storage=$(oc get standardinfraconfig $infra -o
jsonpath='{.spec.broker.resources.storage}')
router_memory=$(oc get standardinfraconfig $infra -o
jsonpath='{.spec.router.resources.memory}')

total_pods=$((routers + brokers + 1))
total_memory_mb=$(( (routers * ${router_memory%Mi}) + (brokers * ${broker_memory%Mi})
+ ${operator_memory%Mi}))
total_storage_gb=$(( brokers * ${broker_storage%Gi}))

echo "Pods: ${total_pods}. Memory: ${total_memory_mb} MB. Storage: ${total_storage_gb}
GB"
```

2. 以下のコマンドを使用してスクリプトを実行します。

```
bash calculate-memory.sh standard-small
```

すべてのコンポーネントに、想定される単位で制限が定義されている場合、スクリプトは、次の例のように、この計画を使用してアドレス空間の合計リソース制限を出力します。

```
Pods: 3. Memory: 1280 MB. Storage: 2 GB
```

8.5. アドレスのサイジング

アドレスブローカーごとのメモリー制限は、アドレスプランの設定から計算されます。AMQ Online は、ブローカー設定の **globalMaxSize** (**standardinfraconfig** または **brokeredinfraconfig** で指定) にアドレスプランのブローカーリソース制限を掛けて、各キューに許可される最大サイズを決定します。キューがメモリー制限に達したときの動作は、アドレスフルポリシーによって管理されます。アドレスフルポリシーの詳細は、ブローカーコンポーネントのサイジング を参照してください。

たとえば、ブローカーの設定で **globalMaxSize** = 124 MB が指定され、アドレスプランの設定で **addressplan.spec.resources.broker** = 0.2 が指定されている場合、各キューに許可される最大サイズは 25 MB (**124 * 0.2 = 25 MB**) です。

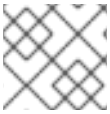
第9章 AMQ ONLINE のリソース設定について

9.1. AMQ ONLINE のアドレス空間とアドレスの概念

AMQ Online のリソースの設定を開始する前に、まずアドレス空間と AMQ Online のアドレスの概念を理解する必要があります。

9.1.1. アドレス空間

アドレス空間は、1 つの接続 (プロトコルごと) を介してアクセスできるアドレスのグループです。アドレス空間のエンドポイントに接続されたクライアントは、そのアドレス空間内の承認されたアドレスとの間でメッセージを送受信できます。アドレス空間は、アドレス空間タイプで定義されているように、複数のプロトコルをサポートできます。



注記

既存のアドレス空間のエンドポイントは、変更できません。

AMQ Online には、次の 2 種類のアドレス空間があります。

- Standard (標準)
- Brokered (ブローカー)

9.1.2. アドレス

アドレスは、アドレス空間の一部であり、メッセージを送受信するための宛先を表します。アドレスには、そのアドレスとの間でメッセージを送受信するセマンティクスを定義するタイプがあります。

AMQ Online で使用できるアドレスの種類は、アドレス空間の種類によって異なります。

9.2. サービス設定のリソースと定義

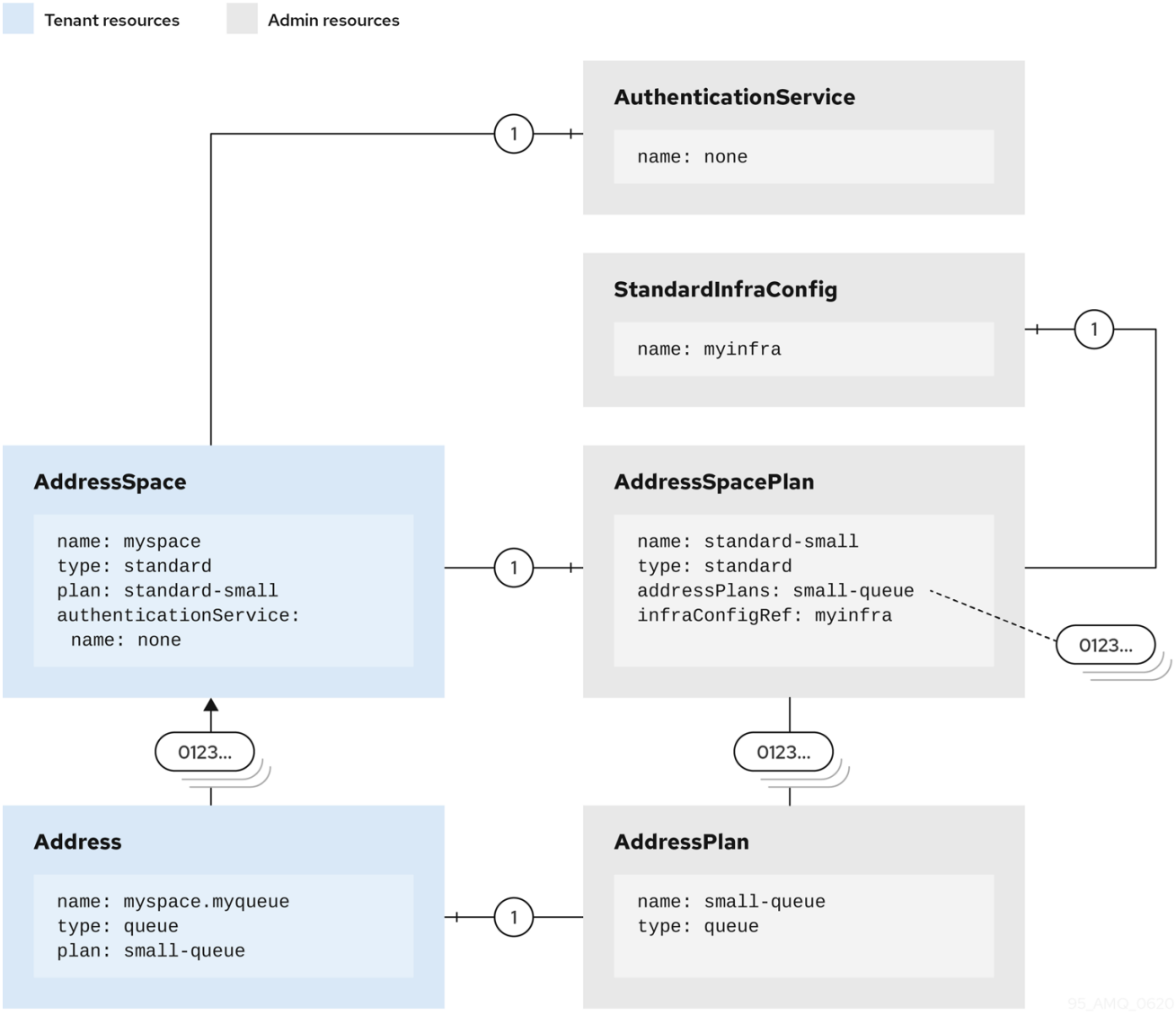
サービス管理者は、「サービス構成」を設定するカスタムリソースを作成して AMQ Online を設定します。 このサービス設定には、次のカスタムリソースタイプのインスタンスが含まれています。

カスタムリソースタイプ	説明
AuthenticationService	メッセージングクライアントの認証に使用する認証サービスインスタンスを指定します。
AddressSpacePlan	使用可能なアドレスプランや、使用できるルーターおよびブローカーリソースの量など、このプランを使用するアドレス空間で使用できるメッセージングリソースを指定します。
AddressPlan	このプランを使用して、特定のアドレスが消費するメッセージングリソースを指定します。これには、アドレスが使用できるルーターとブローカーの割合や、複数のアドレスに指定できるその他のプロパティーなどがあります。

カスタムリソースタイプ	説明
StandardInfraConfig	**standard** アドレス空間タイプの場合は、メモリー制限、ストレージ容量、アフィニティーなど、ルーターとブローカーの設定を指定します。
BrokeredInfraConfig	**brokered** アドレス空間タイプの場合は、メモリー制限、ストレージ容量、アフィニティーなどのブローカー設定を指定します。

これらのカスタムリソースを作成すると、メッセージングテナントが使用できる設定が定義されます。

次の図は、さまざまなサービス設定リソース間の関係と、それらがメッセージングテナントリソースによってどのように参照されるかを示しています。



9.3. AMQ ONLINE を設定するためのユースケースの例

特定のユースケースを満たすためにサービス設定リソースを定義する方法を説明するために、AMQ Online を使用するための X 社の要件を概説します。このユースケースは、サービス設定のリソースタイプをさらに詳しく説明する次のドキュメント全体で参照されます。

X 社には次の要件があります。

- エンジニアリングや品質保証 (QA) の作業チームなど、メッセージングを個別に使用する複数の個別のチームに対応する機能。この要件を満たすには、複数のアドレス空間が必要です。
- X 社のアプリケーションは、JMS API を使用し、ローカルトランザクションを広範囲に使用するように作成されており、AMQP クライアントと OpenWire クライアントを組み合わせて使用するため、仲介アドレス空間タイプを使用する必要があります。
- エンジニアリング作業では、最大 10 個のキューとトピックを使用して、1 メッセージあたり約 1 KB の最大 1000 メッセージのストレージをサポートするようにメッセージングインフラストラクチャーを制限する必要があります。
  QA 作業では、最大 50 個のキューとトピックを使用して、約 100 KB のメッセージを 10,000 個以下のストレージをサポートするようにメッセージングインフラストラクチャーを制限する必要があります。
- エンジニアリング作業では、アドレス空間に接続できるユーザーを制限する機能が必要です。
- エンジニアリング作業のために、エンジニアリングチームは、個別に認証する必要がある個別のユーザーを作成する必要はありません。
  QA 作業のために、QA チームはインスタンスごとにユーザーを作成できる必要があります。

これらの各要件と、適切なリソースを設定することで要件を満たす方法については、次のセクションで説明します。

9.3.1. メッセージングインフラストラクチャーの制限

X 社には、AMQ Online を使用するための次の要件があります。

- エンジニアリング作業では、最大 10 個のキューとトピックを使用して、1 メッセージあたり約 1 KB の最大 1000 メッセージのストレージをサポートするようにメッセージングインフラストラクチャーを制限する必要があります。
  QA 作業では、最大 50 個のキューとトピックを使用して、約 100 KB のメッセージを 10,000 個以下のストレージをサポートするようにメッセージングインフラストラクチャーを制限する必要があります。

この要件を満たすには、**BrokeredInfraConfig** リソースを設定する必要があります。次の点を考慮する必要があります。

- ブローカのメモリーサイズを計算する: 要件を考えると、エンジニアリング作業には比較的小さなメモリーサイズを指定するだけで十分ですが、QA 作業にはより多くのメモリーが必要です。ブローカーのサイジングガイドラインの詳細は、ブローカーコンポーネントのサイジング を参照してください。
- ブローカーのストレージの最小量を計算します。ブローカーのサイジングガイドラインの詳細は、ブローカーコンポーネントのサイジング を参照してください。

9.3.1.1. 仲介インフラ設定の例

次の仲介インフラストラクチャーの設定例は、X 社の要件を満たす仲介コンポーネントのリソース値を示しています。

エンジニアリング向け仲介インフラ設定例

```
apiVersion: admin.enmasse.io/v1beta1
kind: BrokeredInfraConfig
metadata:
  name: engineering
spec:
  broker:
    resources:
      memory: 512Mi
      storage: 20Mi
```

QA 用仲介インフラストラクチャー設定例

```
apiVersion: admin.enmasse.io/v1beta1
kind: BrokeredInfraConfig
metadata:
  name: qa
spec:
  broker:
    resources:
      memory: 4Gi
      storage: 50Gi
```

9.3.2. アドレス空間接続を制限する機能

X 社には、AMQ Online を使用するための次の要件があります。エンジニアリング作業のために、アドレス空間に接続できるユーザーを制限する機能が必要です。

この要件を満たすには、仲介インフラストラクチャー設定でネットワークポリシーを設定する必要があります。ネットワークポリシーの詳細は、次を参照してください。

- ネットワークポリシーの有効化 に関する OpenShift Container Platform 3.11 ドキュメント。
- OpenShift SDN を使用したネットワークポリシーの設定 に関する OpenShift Container Platform 4.2 ドキュメント。

ネットワークポリシー設定を示す仲介インフラストラクチャーの設定例

```
apiVersion: admin.enmasse.io/v1beta1
kind: BrokeredInfraConfig
metadata:
  name: engineering
spec:
  networkPolicy:
    ingress:
      - from:
        - namespaceSelector:
            matchLabels:
              org: engineering
  broker:
    resources:
      memory: 512Mi
      storage: 20Mi
```

さらに、アドレス空間プランは以前の **BrokeredInfraConfig** カスタムリソースを参照します。

アドレス空間計画の例

```
apiVersion: admin.enmasse.io/v1beta2
kind: AddressSpacePlan
metadata:
  name: engineering
spec:
  infraConfigRef: engineering
  addressSpaceType: brokered
  addressPlans:
  - brokered-queue
  - brokered-topic
```

9.3.3. 認証サービスリソースの例

会社 X には、AMQ Online を使用するための次の要件があります。エンジニアリング作業のために、エンジニアリングチームは、個別に認証する必要がある個別のユーザーを作成する必要はありません。この要件を満たすには、**none** 認証サービスを指定します。

認証なしサービスの例

```
apiVersion: admin.enmasse.io/v1beta1
kind: AuthenticationService
metadata:
  name: engineering
spec:
  type: none
```

QA 作業のために、QA チームはインスタンスごとにユーザーを作成できる必要があります。また、QA には、ユーザーを永続化するために使用するデータベースがあります。この要件を満たすには、**standard** 認証サービスを使用し、データソースを指定する必要があります。

標準認証サービスの例

```
apiVersion: admin.enmasse.io/v1beta1
kind: AuthenticationService
metadata:
  name: qa
spec:
  type: standard
  standard:
    storage:
      type: persistent-claim
      size: 5Gi
    datasource:
      type: postgresql
      host: db.example.com
      port: 5432
      database: authdb
```

付録A サービス管理者向け AMQ ONLINE リソース

次の表では、サービス管理者のロールに関連する AMQ Online リソースについて説明します。

表A.1 AMQ Online サービス管理者リソーステーブル

リソース	説明
addressplans	アドレス計画を指定します。
addressspaceplans	アドレス空間計画を指定します。
addressspaceschemas	**addresspace** で利用できるサービス特性を定義します。**addressspace** は、1 つの **addressspaceschema** を参照します。**standard** と **brokered** は、事前定義された **addressspaceschemas** です。
brokeredinfraconfigs	仲介アドレス空間のインフラストラクチャー設定を指定します。詳細は、仲介インフラストラクチャーの設定フィールドの表 を参照してください。
standardinfraconfigs	標準アドレス空間のインフラストラクチャー設定を指定します。詳細は、標準インフラストラクチャー設定フィールドの表 を参照してください。

付録B 仲介インフラ設定フィールド

この表は、仲介インフラストラクチャー設定で使用可能なフィールドと簡単な説明を示しています。

表B.1 仲介インフラ設定フィールドの表

フィールド	説明
version	使用する AMQ Online バージョンを指定します。アップグレード時に、AMQ Online はこのフィールドを使用して、インフラストラクチャーを要求されたバージョンにアップグレードするかどうかを決定します。
admin.resources.memory	管理 Pod に割り当てられるメモリーの量を指定します。
admin.podTemplate.metadata.labels	管理 Pod に追加されるラベルを指定します。
admin.podTemplate.spec.affinity	管理 Pod のアフィニティー設定を指定して、特定のノードで Pod を実行する場所を指定したり、他のインスタンスと一緒に実行できないかどうかを指定したりできます。
admin.podTemplate.spec.priorityClassName	OpenShift クラスター内の他の Pod よりも管理 Pod を優先できるように、管理 Pod に使用する優先順位クラスを指定します。
admin.podTemplate.spec.tolerations	管理 Pod の容認設定を指定します。これにより、他の Pod が実行できない特定のノードでこの Pod を実行できるようになります。
broker.addressFullPolicy	キューがいっぱいになったときに実行するアクションを指定します (**BLOCK**、**FAIL**、**PAGE**、**DROP**)。デフォルト値は **PAGE** です。詳細は、AMQ Broker のドキュメント を参照してください。
broker.globalMaxSize	ブローカのキューに使用されるメモリーの最大量を指定します。
broker.resources.memory	ブローカーに割り当てられるメモリーの量を指定します。
broker.resources.storage	ブローカーに要求されたストレージの量を指定します。
broker.podTemplate.metadata.labels	ブローカー Pod に追加されるラベルを指定します。

broker.podTemplate.spec.affinity	ブローカー Pod のアフィニティー設定を指定して、特定のノードで Pod を実行する場所を指定したり、他のインスタンスと一緒に実行できないかどうかを指定したりできます。
broker.podTemplate.spec.priorityClassName	ブローカー Pod に使用する優先順位クラスを指定して、OpenShift クラスター内の他の Pod よりもブローカー Pod を優先できるようにします。
broker.podTemplate.spec.tolerations	ブローカー Pod の許容設定を指定します。これにより、他の Pod が実行できない特定のノードでこの Pod を実行できるようになります。
broker.podTemplate.spec.securityContext	ブローカー Pod の セキュリティーコンテキスト を指定します。
broker.podTemplate.spec.containers.env	ブローカー Pod の環境変数を指定します。
broker.podTemplate.spec.containers.livenessProbe.failureThreshold	コンテナーを再起動する前に、ブローカー Pod が起動し、プローブが失敗したときに、OpenShift が試行する回数を指定します。
broker.podTemplate.spec.containers.livenessProbe.initialDelaySeconds	ブローカー Pod のプローブ遅延値を秒単位で指定します。
broker.podTemplate.spec.containers.livenessProbe.timeoutSeconds	ブローカー Pod のプローブタイムアウト値を秒単位で指定します。
broker.podTemplate.spec.containers.readinessProbe.failureThreshold	ブローカー Pod が起動し、Pod が **Unready** とマークされる前にプローブが失敗したときに、OpenShift が試行する回数を指定します。
broker.podTemplate.spec.containers.readinessProbe.initialDelaySeconds	ブローカー Pod のプローブ遅延値を秒単位で指定します。
broker.podTemplate.spec.containers.readinessProbe.timeoutSeconds	ブローカー Pod のプローブタイムアウト値を秒単位で指定します。
broker.podTemplate.spec.containers.resources	CPU とメモリーのブローカー Pod リソースリクエストと制限を指定します。
broker.storageClassName	ブローカーの永続ボリュームに使用するストレージクラスを指定します。
broker.updatePersistentVolumeClaim	永続ボリュームがサイズ変更をサポートしている場合は、この値を **true** に設定すると、ブローカーストレージのサイズを変更できます。

付録C 標準インフラストラクチャー設定フィールド

この表は、標準インフラストラクチャー設定で使用可能なフィールドと簡単な説明を示しています。

表C.1標準インフラストラクチャー設定フィールドの表

フィールド	説明
version	使用する AMQ Online バージョンを指定します。アップグレード時に、AMQ Online はこのフィールドを使用して、インフラストラクチャーを要求されたバージョンにアップグレードするかどうかを決定します。
admin.resources.memory	管理 Pod に割り当てられるメモリーの量を指定します。
admin.podTemplate.metadata.labels	管理 Pod に追加されるラベルを指定します。
admin.podTemplate.spec.affinity	管理 Pod のアフィニティー設定を指定して、特定のノードで Pod を実行する場所を指定したり、他のインスタンスと一緒に実行できないかどうかを指定したりできます。
admin.podTemplate.spec.priorityClassName	OpenShift クラスター内の他の Pod よりも管理 Pod を優先できるように、管理 Pod に使用する優先度クラスを指定します。
admin.podTemplate.spec.tolerations	管理 Pod の容認設定を指定します。これにより、他の Pod が実行できない特定のノードでこの Pod を実行できるようになります。
broker.addressFullPolicy	キューがいっぱいになったときに実行するアクションを指定します (**BLOCK**、**FAIL**、**PAGE**、**DROP**)。デフォルト値は **PAGE** です。詳細は、AMQ Broker のドキュメント を参照してください。
broker.globalMaxSize	ブローカのキューに使用されるメモリーの最大量を指定します。
broker.resources.memory	ブローカーに割り当てられるメモリーの量を指定します。
broker.resources.storage	ブローカーに要求されたストレージの量を指定します。
broker.podTemplate.metadata.labels	ブローカー Pod に追加されるラベルを指定します。

broker.podTemplate.spec.affinity	ブローカー Pod のアフィニティー設定を指定して、特定のノードで Pod を実行する場所を指定したり、他のインスタンスと一緒に実行できないかどうかを指定したりできます。
broker.podTemplate.spec.priorityClassName	ブローカー Pod に使用する優先順位クラスを指定して、OpenShift クラスター内の他の Pod よりもブローカー Pod を優先できるようにします。
broker.podTemplate.spec.tolerations	ブローカー Pod の許容設定を指定します。これにより、他の Pod が実行できない特定のノードでこの Pod を実行できるようになります。
broker.podTemplate.spec.securityContext	ブローカー Pod の セキュリティーコンテキスト を指定します。
broker.podTemplate.spec.containers.env	ブローカー Pod の環境変数を指定します。
broker.podTemplate.spec.containers.livenessProbe.failureThreshold	コンテナーを再起動する前に、ブローカー Pod が起動し、プローブが失敗したときに、OpenShift が試行する回数を指定します。
broker.podTemplate.spec.containers.livenessProbe.initialDelaySeconds	ブローカー Pod のプローブ遅延値を秒単位で指定します。
broker.podTemplate.spec.containers.livenessProbe.timeoutSeconds	ブローカー Pod のプローブタイムアウト値を秒単位で指定します。
broker.podTemplate.spec.containers.readinessProbe.failureThreshold	ブローカー Pod が起動し、Pod が **Unready** とマークされる前にプローブが失敗したときに、OpenShift が試行する回数を指定します。
broker.podTemplate.spec.containers.readinessProbe.initialDelaySeconds	ブローカー Pod のプローブ遅延値を秒単位で指定します。
broker.podTemplate.spec.containers.readinessProbe.timeoutSeconds	ブローカー Pod のプローブタイムアウト値を秒単位で指定します。
broker.podTemplate.spec.containers.resources	CPU とメモリーのブローカー Pod リソースリクエストと制限を指定します。
broker.connectorIdleTimeout	ルーターへの接続に使用する AMQP アイドルタイムアウトを指定します。
broker.connectorWorkerThreads	ルーターへの接続のワーカースレッドの数を指定します。

broker.storageClassName	ブローカーの永続ボリュームに使用するストレージクラスを指定します。
broker.updatePersistentVolumeClaim	永続ボリュームがサイズ変更をサポートしている場合は、この値を **true** に設定すると、ブローカーストレージのサイズを変更できます。
broker.treatRejectAsUnmodifiedDeliveryFailed	変更された配信の失敗として、拒否された配信結果を扱います。これにより、デフォルトでメッセージがコンシューマーに再送信されます。デフォルト値は **true** です。
broker.useModifiedForTransientDeliveryErrors	送信者が再試行できるように、一時的な配信エラーに対して変更された応答を返します。デフォルト値は **true** です。
broker.minLargeMessageSize	大きなメッセージとして扱われるメッセージの最小サイズを指定します。サイズの大きなメッセージは、ジャーナル内の参照を使用して常にディスクにページングされます。デフォルト値は **-1** (ページングしない) です。
router.resources.memory	ルーターに割り当てられるメモリーの量を指定します。
router.linkCapacity	ルーターの AMQP リンクで発行されるデフォルトのクレジット数を指定します。
router.handshakeTimeout	安全なハンドシェークが開始されるまで待機する時間を秒単位で指定します。
router.minReplicas	実行するルーター Pod の最小数を指定します。高可用性 (HA) 設定には、少なくとも 2 つが必要です。
router.podTemplate.metadata.labels	ルーター Pod に追加されるラベルを指定します。
router.podTemplate.spec.affinity	ルーター Pod のアフィニティー設定を指定して、特定のノードで Pod を実行する場所を指定したり、他のインスタンスと一緒に実行できないかどうかを指定したりできます。
router.podTemplate.spec.priorityClassName	ルーター Pod に使用する優先度クラスを指定して、OpenShift クラスター内の他の Pod よりもルーター Pod に優先順位を高く付けることができるようにします。
router.podTemplate.spec.tolerations	ルーター Pod の許容設定を指定します。これにより、他の Pod が実行できない特定のノードでこの Pod を実行できるようになります。

broker.podTemplate.spec.securityContext	ルーター Pod の セキュリティーコンテキスト を指定します。
router.podTemplate.spec.containers.env	ルーター Pod の環境変数を指定します。
router.podTemplate.spec.containers.livenessProbe.failureThreshold	コンテナーを再起動する前に、ルーター Pod が起動してプローブが失敗したときに、OpenShift が試行する回数を指定します。
router.podTemplate.spec.containers.livenessProbe.initialDelaySeconds	ルーター Pod のプローブ遅延値を秒単位で指定します。
router.podTemplate.spec.containers.livenessProbe.timeoutSeconds	ルーター Pod のプローブタイムアウト値を秒単位で指定します。
router.podTemplate.spec.containers.readinessProbe.failureThreshold	ルーター Pod が起動し、Pod が **Unready** とマークされる前にプローブが失敗したときに、OpenShift が試行する回数を指定します。
router.podTemplate.spec.containers.readinessProbe.initialDelaySeconds	ルーター Pod のプローブ遅延値を秒単位で指定します。
router.podTemplate.spec.containers.readinessProbe.timeoutSeconds	ルーター Pod のプローブタイムアウト値を秒単位で指定します。
router.podTemplate.spec.containers.resources	ルーター Pod のリソースリクエストと、CPU とメモリーの制限を指定します。
router.idleTimeout	すべてのルーターリスナーに使用する AMQP アイドルタイムアウトを指定します。
router.workerThreads	ルーターに使用するワーカースレッドの数を指定します。
router.policy.maxConnections	許可されるルーター接続の最大数を指定します。
router.policy.maxConnectionsPerUser	ユーザーごとに許可されるルーター接続の最大数を指定します。
router.policy.maxConnectionsPerHost	ホストごとに許可されるルーター接続の最大数を指定します。
router.policy.maxSessionsPerConnection	ルーター接続ごとに許可されるセッションの最大数を指定します。
router.policy.maxSendersPerConnection	ルーター接続ごとに許可される送信者の最大数を指定します。

router.policy.maxReceiversPerConnection	ルーター接続ごとに許可される受信者の最大数を指定します。

付録D REST API リファレンス

D.1. ENMASSE REST API

D.1.1. 概要

これは EnMasse API 仕様です。

D.1.1.1. バージョン情報

バージョン : 0.32-SNAPSHOT

D.1.1.2. URI スキーム

Schemes: HTTPS

D.1.1.3. タグ

- addresses : アドレスに対する操作。
- addressplans : AddressPlans に対する操作。
- addressspaceplans : AddressSpacePlans に対する操作。
- addressspaces : AddressSpaces に対する操作。
- brokeredinfraconfigs : BrokeredInfraConfigs に対する操作。
- messagingusers : MessagingUsers に対する操作。
- standardinfraconfigs : StandardInfraConfigs に対する操作。

D.1.1.4. 外部ドキュメント

説明: EnMasse について詳しく知る
URL: https://enmasse.io/documentation/

D.1.2. パス

D.1.2.1. POST /apis/admin.enmasse.io/v1beta2/namespaces/{namespace}/addressspaceplans

D.1.2.1.1. 説明

AddressSpacePlan の作成

D.1.2.1.2. パラメーター

タイプ	Name	説明	スキーマ

タイプ	Name	説明	スキーマ
パス	**namespace** **必須**	チームやプロジェクトなどのオブジェクト名と認証スコープ	string
本文	**body** **必須**		io.enmasse.admin.v1beta2.AddressSpacePlan

D.1.2.1.3. 応答

HTTP コード	説明	スキーマ
200	OK	io.enmasse.admin.v1beta2.AddressSpacePlan
201	Created	io.enmasse.admin.v1beta2.AddressSpacePlan
401	認可されていない	コンテンツなし

D.1.2.1.4. 消費

- **application/json**

D.1.2.1.5. 生成されるアイテム

- **application/json**

D.1.2.1.6. タグ

- addressspaceplan
- admin
- enmasse_v1beta2

D.1.2.2. GET /apis/admin.enmasse.io/v1beta2/namespaces/{namespace}/addressspaceplans

D.1.2.2.1. 説明

種類 AddressSpacePlan のオブジェクトの一覧表示

D.1.2.2.2. パラメーター

タイプ	Name	説明	スキーマ
パス	**namespace** **必須**	チームやプロジェクトなどのオブジェクト名と認証スコープ	string
クエリー	**labelSelector** **任意**	返されるオブジェクトのリストをラベルで制限するためのセレクターです。デフォルトはすべてです。	string

D.1.2.2.3. 応答

HTTP コード	説明	スキーマ
200	OK	io.enmasse.admin.v1beta2.AddressSpacePlanList
401	認可されていない	コンテンツなし

D.1.2.2.4. 生成されるアイテム

- **application/json**

D.1.2.2.5. タグ

- addressspaceplan
- admin
- enmasse_v1beta2

D.1.2.3. GET /apis/admin.enmasse.io/v1beta2/namespaces/{namespace}/addressspaceplans/{name}

D.1.2.3.1. 説明

指定された AddressSpacePlan を読み取り

D.1.2.3.2. パラメーター

タイプ	Name	説明	スキーマ
パス	**name** **必須**	読み取る AddressSpacePlan の名前。	string
パス	**namespace** **必須**	チームやプロジェクトなどのオブジェクト名と認証スコープ	string

D.1.2.3.3. 応答

HTTP コード	説明	スキーマ
200	OK	io.enmasse.admin.v1beta2.AddressSpacePlan
401	認可されていない	コンテンツなし
404	見つからない	コンテンツなし

D.1.2.3.4. 消費

- **application/json**

D.1.2.3.5. 生成されるアイテム

- **application/json**

D.1.2.3.6. タグ

- addressspaceplan
- admin
- enmasse_v1beta2

D.1.2.4. PUT /apis/admin.enmasse.io/v1beta2/namespaces/{namespace}/addressspaceplans/{name}

D.1.2.4.1. 説明

指定された AddressSpacePlan を置き換え

D.1.2.4.2. パラメーター

タイプ	Name	説明	スキーマ
パス	**name** **必須**	置き換える AddressSpacePlan の名前。	string
パス	**namespace** **必須**	チームやプロジェクトなどのオブジェクト名と認証スコープ	string
本文	**body** **必須**		io.enmasse.admin.v1beta2.AddressSpacePlan

D.1.2.4.3. 応答

HTTP コード	説明	スキーマ
200	OK	io.enmasse.admin.v1beta2.AddressSpacePlan
201	Created	io.enmasse.admin.v1beta2.AddressSpacePlan
401	認可されていない	コンテンツなし

D.1.2.4.4. 生成されるアイテム

- **application/json**

D.1.2.4.5. タグ

- addressspaceplan
- admin
- enmasse_v1beta2

D.1.2.5. DELETE /apis/admin.enmasse.io/v1beta2/namespaces/{namespace}/addressspaceplans/{name}

D.1.2.5.1. 説明

AddressSpacePlan を削除する

D.1.2.5.2. パラメーター

タイプ	Name	説明	スキーマ
パス	**name** **必須**	削除する AddressSpacePlan の名前。	string
パス	**namespace** **必須**	チームやプロジェクトなどのオブジェクト名と認証スコープ	string

D.1.2.5.3. 応答

HTTP コード	説明	スキーマ
200	OK	Status
401	認可されていない	コンテンツなし
404	見つからない	コンテンツなし

D.1.2.5.4. 生成されるアイテム

- **application/json**

D.1.2.5.5. タグ

- addressspaceplan
- admin
- enmasse_v1beta2

D.1.2.6. POST /apis/enmasse.io/v1beta1/namespaces/{namespace}/addresses

D.1.2.6.1. 説明

アドレスの作成

D.1.2.6.2. パラメーター

タイプ	Name	説明	スキーマ
パス	**namespace** **必須**	チームやプロジェクトなどのオブジェクト名と認証スコープ	string
本文	**body** **必須**		io.enmasse.v1beta1.Address

D.1.2.6.3. 応答

HTTP コード	説明	スキーマ
200	OK	io.enmasse.v1beta1.Address
201	Created	io.enmasse.v1beta1.Address

HTTP コード	説明	スキーマ
401	認可されていない	コンテンツなし

D.1.2.6.4. 消費

- **application/json**

D.1.2.6.5. 生成されるアイテム

- **application/json**

D.1.2.6.6. タグ

- addresses
- enmasse_v1beta1

D.1.2.7. GET /apis/enmasse.io/v1beta1/namespaces/{namespace}/addresses

D.1.2.7.1. 説明

種類 Address のオブジェクトの一覧表示

D.1.2.7.2. パラメーター

タイプ	Name	説明	スキーマ
パス	**namespace 必須**	チームやプロジェクトなどのオブジェクト名と認証スコープ	string
クエリー	**labelSelector 任意**	返されるオブジェクトのリストをラベルで制限するためのセレクターです。デフォルトはすべてです。	string

D.1.2.7.3. 応答

HTTP コード	説明	スキーマ
200	OK	io.enmasse.v1beta1.AddressList
401	認可されていない	コンテンツなし

D.1.2.7.4. 生成されるアイテム

- **application/json**

D.1.2.7.5. タグ

- addresses
- enmasse_v1beta1

D.1.2.8. GET /apis/enmasse.io/v1beta1/namespaces/{namespace}/addresses/{name}

D.1.2.8.1. 説明

指定されたアドレスを読み取り

D.1.2.8.2. パラメーター

タイプ	Name	説明	スキーマ
パス	**name** **必須**	読む住所の名前	string
パス	**namespace** **必須**	チームやプロジェクトなどのオブジェクト名と認証スコープ	string

D.1.2.8.3. 応答

HTTP コード	説明	スキーマ
200	OK	io.enmasse.v1beta1.Address
401	認可されていない	コンテンツなし
404	見つからない	コンテンツなし

D.1.2.8.4. 消費

- **application/json**

D.1.2.8.5. 生成されるアイテム

- **application/json**

D.1.2.8.6. タグ

- addresses
- enmasse_v1beta1

D.1.2.9. PUT /apis/enmasse.io/v1beta1/namespaces/{namespace}/addresses/{name}

D.1.2.9.1. 説明

指定されたアドレスを置き換え

D.1.2.9.2. パラメーター

タイプ	Name	説明	スキーマ
パス	**name** **必須**	置換する住所の名前	string
パス	**namespace** **必須**	チームやプロジェクトなどのオブジェクト名と認証スコープ	string
本文	**body** **必須**		io.enmasse.v1beta1.Address

D.1.2.9.3. 応答

HTTP コード	説明	スキーマ
200	OK	io.enmasse.v1beta1.Address
201	Created	io.enmasse.v1beta1.Address
401	認可されていない	コンテンツなし

D.1.2.9.4. 生成されるアイテム

- **application/json**

D.1.2.9.5. タグ

- addresses
- enmasse_v1beta1

D.1.2.10. DELETE /apis/enmasse.io/v1beta1/namespaces/{namespace}/addresses/{name}

D.1.2.10.1. 説明

アドレスを削除する

D.1.2.10.2. パラメーター

タイプ	Name	説明	スキーマ
パス	**name** **必須**	削除するアドレスの名前	string
パス	**namespace** **必須**	チームやプロジェクトなどのオブジェクト名と認証スコープ	string

D.1.2.10.3. 応答

HTTP コード	説明	スキーマ
200	OK	Status
401	認可されていない	コンテンツなし
404	見つからない	コンテンツなし

D.1.2.10.4. 生成されるアイテム

- **application/json**

D.1.2.10.5. タグ

- addresses
- enmasse_v1beta1

D.1.2.11. PATCH /apis/enmasse.io/v1beta1/namespaces/{namespace}/addresses/{name}

D.1.2.11.1. 説明

指定されたアドレスにパッチを適用 (RFC6902)

D.1.2.11.2. パラメーター

タイプ	Name	説明	スキーマ
パス	**name** **必須**	置換する住所の名前	string
パス	**namespace** **必須**	チームやプロジェクトなどのオブジェクト名と認証スコープ	string
本文	**body** **必須**		JsonPatchRequest

D.1.2.11.3. 応答

HTTP コード	説明	スキーマ
200	OK	io.enmasse.v1beta1.Address
401	認可されていない	コンテンツなし

D.1.2.11.4. 消費されるアイテム

- **application/json-patch+json**

D.1.2.11.5. 生成されるアイテム

- **application/json**

D.1.2.11.6. タグ

- addresses
- enmasse_v1beta1

D.1.2.12. POST /apis/enmasse.io/v1beta1/namespaces/{namespace}/addressspaces

D.1.2.12.1. 説明

AddressSpace の作成

D.1.2.12.2. パラメーター

タイプ	Name	説明	スキーマ
パス	**namespace** **必須**	チームやプロジェクトなどのオブジェクト名と認証スコープ	string
本文	**body** **必須**		io.enmasse.v1beta1.AddressSpace

D.1.2.12.3. 応答

HTTP コード	説明	スキーマ
200	OK	io.enmasse.v1beta1.AddressSpace

HTTP コード	説明	スキーマ
201	Created	io.enmasse.v1beta1.AddressSpace
401	認可されていない	コンテンツなし

D.1.2.12.4. 消費

- **application/json**

D.1.2.12.5. 生成されるアイテム

- **application/json**

D.1.2.12.6. タグ

- addressspaces
- enmasse_v1beta1

D.1.2.13. GET /apis/enmasse.io/v1beta1/namespaces/{namespace}/addressspaces

D.1.2.13.1. 説明

種類が AddressSpace のオブジェクトの一覧表示

D.1.2.13.2. パラメーター

タイプ	Name	説明	スキーマ
パス	**namespace 必須**	チームやプロジェクトなどのオブジェクト名と認証スコープ	string
クエリー	**labelSelector 任意**	返されるオブジェクトのリストをラベルで制限するためのセレクターです。デフォルトはすべてです。	string

D.1.2.13.3. 応答

HTTP コード	説明	スキーマ
200	OK	io.enmasse.v1beta1.AddressSpaceList
401	認可されていない	コンテンツなし

D.1.2.13.4. 生成されるアイテム

- **application/json**

D.1.2.13.5. タグ

- addressspaces
- enmasse_v1beta1

D.1.2.14. GET /apis/enmasse.io/v1beta1/namespaces/{namespace}/addressspaces/{name}

D.1.2.14.1. 説明

指定された AddressSpace の読み取り

D.1.2.14.2. パラメーター

タイプ	Name	説明	スキーマ
パス	**name** **必須**	読み取る AddressSpace の名前	string
パス	**namespace** **必須**	チームやプロジェクトなどのオブジェクト名と認証スコープ	string

D.1.2.14.3. 応答

HTTP コード	説明	スキーマ
200	OK	io.enmasse.v1beta1.AddressSpace
401	認可されていない	コンテンツなし
404	見つからない	コンテンツなし

D.1.2.14.4. 消費

- **application/json**

D.1.2.14.5. 生成されるアイテム

- **application/json**

D.1.2.14.6. タグ

- addressspaces

- enmasse_v1beta1

D.1.2.15. PUT /apis/enmasse.io/v1beta1/namespaces/{namespace}/addressspaces/{name}

D.1.2.15.1. 説明

指定された AddressSpace の置き換え

D.1.2.15.2. パラメーター

タイプ	Name	説明	スキーマ
パス	**name** **必須**	置き換える AddressSpace の名前	string
パス	**namespace** **必須**	チームやプロジェクトなどのオブジェクト名と認証スコープ	string
本文	**body** **必須**		io.enmasse.v1beta1.AddressSpace

D.1.2.15.3. 応答

HTTP コード	説明	スキーマ
200	OK	io.enmasse.v1beta1.AddressSpace
201	Created	io.enmasse.v1beta1.AddressSpace
401	認可されていない	コンテンツなし

D.1.2.15.4. 生成されるアイテム

- **application/json**

D.1.2.15.5. タグ

- addressspaces
- enmasse_v1beta1

D.1.2.16. DELETE /apis/enmasse.io/v1beta1/namespaces/{namespace}/addressspaces/{name}

D.1.2.16.1. 説明

AddressSpace の削除

D.1.2.16.2. パラメーター

タイプ	Name	説明	スキーマ
パス	**name** **必須**	削除する AddressSpace の名前	string
パス	**namespace** **必須**	チームやプロジェクトなどのオブジェクト名と認証スコープ	string

D.1.2.16.3. 応答

HTTP コード	説明	スキーマ
200	OK	Status
401	認可されていない	コンテンツなし
404	見つからない	コンテンツなし

D.1.2.16.4. 生成されるアイテム

- **application/json**

D.1.2.16.5. タグ

- addressspaces
- enmasse_v1beta1

D.1.2.17. PATCH /apis/enmasse.io/v1beta1/namespaces/{namespace}/addressspaces/{name}

D.1.2.17.1. 説明

指定された AddressSpace にパッチを適用 (RFC6902)

D.1.2.17.2. パラメーター

タイプ	Name	説明	スキーマ
パス	**name** **必須**	置き換える AddressSpace の名前	string

タイプ	Name	説明	スキーマ
パス	**namespace** **必須**	チームやプロジェクトなどのオブジェクト名と認証スコープ	string
本文	**body** **必須**		JsonPatchRequest

D.1.2.17.3. 応答

HTTP コード	説明	スキーマ
200	OK	io.enmasse.v1beta1.AddressSpace
401	認可されていない	コンテンツなし

D.1.2.17.4. 消費されるアイテム

- **application/json-patch+json**

D.1.2.17.5. 生成されるアイテム

- **application/json**

D.1.2.17.6. タグ

- addressspaces
- enmasse_v1beta1

D.1.2.18. POST /apis/user.enmasse.io/v1beta1/namespaces/{namespace}/messagingusers

D.1.2.18.1. 説明

MessagingUser の作成

D.1.2.18.2. パラメーター

タイプ	Name	説明	スキーマ
パス	**namespace** **必須**	チームやプロジェクトなどのオブジェクト名と認証スコープ	string
本文	**body** **必須**		io.enmasse.user.v1beta1.MessagingUser

D.1.2.18.3. 応答

HTTP コード	説明	スキーマ
200	OK	io.enmasse.user.v1beta1.MessagingUser
201	Created	io.enmasse.user.v1beta1.MessagingUser
401	認可されていない	コンテンツなし

D.1.2.18.4. 消費

- **application/json**

D.1.2.18.5. 生成されるアイテム

- **application/json**

D.1.2.18.6. タグ

- auth
- enmasse_v1beta1
- user

D.1.2.19. GET /apis/user.enmasse.io/v1beta1/namespaces/{namespace}/messagingusers

D.1.2.19.1. 説明

種類 MessagingUser のオブジェクトの一覧表示

D.1.2.19.2. パラメーター

タイプ	Name	説明	スキーマ
パス	**namespace 必須**	チームやプロジェクトなどのオブジェクト名と認証スコープ	string
クエリー	**labelSelector 任意**	返されるオブジェクトのリストをラベルで制限するためのセレクターです。デフォルトはすべてです。	string

D.1.2.19.3. 応答

HTTP コード	説明	スキーマ
200	OK	io.enmasse.user.v1beta1.MessagingUserList
401	認可されていない	コンテンツなし

D.1.2.19.4. 生成されるアイテム

- **application/json**

D.1.2.19.5. タグ

- auth
- enmasse_v1beta1
- user

D.1.2.20. GET /apis/user.enmasse.io/v1beta1/namespaces/{namespace}/messagingusers/{name}

D.1.2.20.1. 説明

指定された MessagingUser の読み取り

D.1.2.20.2. パラメーター

タイプ	Name	説明	スキーマ
パス	**name** **必須**	読み取る MessagingUser の名前。名前に addressSpace とドット区切り文字を含める必要があります (つまり 'myspace.user1')。	string
パス	**namespace** **必須**	チームやプロジェクトなどのオブジェクト名と認証スコープ	string

D.1.2.20.3. 応答

HTTP コード	説明	スキーマ
200	OK	io.enmasse.user.v1beta1.MessagingUser

HTTP コード	説明	スキーマ
401	認可されていない	コンテンツなし
404	見つからない	コンテンツなし

D.1.2.20.4. 消費

- **application/json**

D.1.2.20.5. 生成されるアイテム

- **application/json**

D.1.2.20.6. タグ

- auth
- enmasse_v1beta1
- user

D.1.2.21. PUT /apis/user.enmasse.io/v1beta1/namespaces/{namespace}/messagingusers/{name}

D.1.2.21.1. 説明

指定された MessagingUser の置き換え

D.1.2.21.2. パラメーター

タイプ	Name	説明	スキーマ
パス	**name** **必須**	置き換える MessagingUser の名前。名前に addressSpace とドット区切り文字を含める必要があります (つまり 'myspace.user1')。	string
パス	**namespace** **必須**	チームやプロジェクトなどのオブジェクト名と認証スコープ	string
本文	**body** **必須**		io.enmasse.user.v1beta1.MessagingUser

D.1.2.21.3. 応答

HTTP コード	説明	スキーマ
200	OK	io.enmasse.user.v1beta1.MessagingUser
201	Created	io.enmasse.user.v1beta1.MessagingUser
401	認可されていない	コンテンツなし

D.1.2.21.4. 生成されるアイテム

- **application/json**

D.1.2.21.5. タグ

- auth
- enmasse_v1beta1
- user

D.1.2.22. DELETE /apis/user.enmasse.io/v1beta1/namespaces/{namespace}/messagingusers/{name}

D.1.2.22.1. 説明

MessagingUser の削除

D.1.2.22.2. パラメーター

タイプ	Name	説明	スキーマ
パス	**name** **必須**	削除する MessagingUser の名前。名前に addressSpace とドット区切り文字を含める必要があります (つまり 'myspace.user1')。	string
パス	**namespace** **必須**	チームやプロジェクトなどのオブジェクト名と認証スコープ	string

D.1.2.22.3. 応答

HTTP コード	説明	スキーマ
200	OK	Status
401	認可されていない	コンテンツなし
404	見つからない	コンテンツなし

D.1.2.22.4. 生成されるアイテム

- **application/json**

D.1.2.22.5. タグ

- auth
- enmasse_v1beta1
- user

D.1.2.23. PATCH /apis/user.enmasse.io/v1beta1/namespaces/{namespace}/messagingusers/{name}

D.1.2.23.1. 説明

指定された MessagingUser にパッチを適用 (RFC6902)

D.1.2.23.2. パラメーター

タイプ	Name	説明	スキーマ
パス	**name** **必須**	置き換える MessagingUser の名前。名前に addressSpace とドット区切り記号を含める必要があります (つまり、'myspace.user1')	string
パス	**namespace** **必須**	チームやプロジェクトなどのオブジェクト名と認証スコープ	string
本文	**body** **必須**		JsonPatchRequest

D.1.2.23.3. 応答

HTTP コード	説明	スキーマ

HTTP コード	説明	スキーマ
200	OK	io.enmasse.user.v1beta1.MessagingUser
401	認可されていない	コンテンツなし

D.1.2.23.4. 消費されるアイテム

- **application/json-patch+json**

D.1.2.23.5. 生成されるアイテム

- **application/json**

D.1.2.23.6. タグ

- auth
- enmasse_v1beta1
- user

D.1.3. 定義

D.1.3.1. JsonPatchRequest

名前	スキーマ
document **必須**	object
patch **必須**	< Patch > array

D.1.3.2. ObjectMeta

ObjectMeta は、すべての永続化されたリソースが持つ必要のあるメタデータであり、ユーザーが作成する必要のあるすべてのオブジェクトが含まれます。

名前	スキーマ
name **必須**	string

名前	スキーマ
namespace **任意**	string

D.1.3.3. Patch

名前	説明	スキーマ
from **任意**	オペレーションのコピー、置換に必要	string
op **必須**		列挙型 (追加、削除、置換、移動、コピー、テスト)
path **必須**	スラッシュ区切り形式	string
value **任意**	操作の追加、置換、テストに必要	string

D.1.3.4. Status

status は、他のオブジェクトを返さない呼び出しの戻り値です。

名前	説明	スキーマ
コード **任意**	このステータスの推奨 HTTP 戻りコード。設定されていない場合は 0。	整数 (int32)

D.1.3.5. io.enmasse.admin.v1beta1.BrokeredInfraConfig

名前	スキーマ
apiVersion **必須**	enum (admin.enmasse.io/v1beta1)
kind **必須**	enum (BrokeredInfraConfig)
metadata **必須**	ObjectMeta
spec **必須**	io.enmasse.admin.v1beta1.BrokeredInfraConfigSpec

D.1.3.6. io.enmasse.admin.v1beta1.BrokeredInfraConfigList

名前	スキーマ
apiVersion 必須	enum (admin.enmasse.io/v1beta1)
items 必須	< io.enmasse.admin.v1beta1.BrokeredInfraConfig > array
kind 必須	enum (BrokeredInfraConfigList)

D.1.3.7. io.enmasse.admin.v1beta1.BrokeredInfraConfigSpec

名前	スキーマ
admin 任意	io.enmasse.admin.v1beta1.BrokeredInfraConfigSpecAdmin
broker 任意	io.enmasse.admin.v1beta1.BrokeredInfraConfigSpecBroker
networkPolicy 任意	networkPolicy
version 任意	string

networkPolicy

名前	スキーマ
egress 任意	< io.k8s.api.networking.v1.NetworkPolicyEgressRule > array
ingress 任意	< io.k8s.api.networking.v1.NetworkPolicyIngressRule > array

D.1.3.8. io.enmasse.admin.v1beta1.BrokeredInfraConfigSpecAdmin

名前	スキーマ
podTemplate 任意	io.enmasse.admin.v1beta1.InfraConfigPodSpec

名前	スキーマ
resources 任意	resources

resources

名前	スキーマ
memory 任意	string

D.1.3.9. io.enmasse.admin.v1beta1.BrokeredInfraConfigSpecBroker

名前	スキーマ
addressFullPolicy optional	enum (PAGE, BLOCK, FAIL)
podTemplate 任意	io.enmasse.admin.v1beta1.InfraConfigPodSpec
resources 任意	resources
storageClassName optional	string
updatePersistentVolumeClaim optional	boolean

resources

名前	スキーマ
memory 任意	string
storage 任意	string

D.1.3.10. io.enmasse.admin.v1beta1.InfraConfigPodSpec

名前	スキーマ
metadata **任意**	metadata
spec **任意**	spec

metadata

Name	スキーマ
labels **任意**	object

spec

Name	スキーマ
affinity **任意**	object
containers **任意**	< containers > array
priorityClassName **optional**	string
securityContext **optional**	object
tolerations **任意**	< オブジェクト > 配列

containers

名前	スキーマ
resources **任意**	object

D.1.3.11. io.enmasse.admin.v1beta1.StandardInfraConfig

名前	スキーマ
apiVersion **必須**	enum (admin.enmasse.io/v1beta1)
kind **必須**	enum (StandardInfraConfig)
metadata **必須**	ObjectMeta
spec **必須**	io.enmasse.admin.v1beta1.StandardInfraConfigSpec

D.1.3.12. io.enmasse.admin.v1beta1.StandardInfraConfigList

名前	スキーマ
apiVersion **必須**	enum (admin.enmasse.io/v1beta1)
items **必須**	< io.enmasse.admin.v1beta1.StandardInfraConfig > array
kind **必須**	enum (StandardInfraConfigList)

D.1.3.13. io.enmasse.admin.v1beta1.StandardInfraConfigSpec

名前	スキーマ
admin **任意**	io.enmasse.admin.v1beta1.StandardInfraConfigSpecAdmin
broker **任意**	io.enmasse.admin.v1beta1.StandardInfraConfigSpecBroker
networkPolicy **任意**	networkPolicy
router **任意**	io.enmasse.admin.v1beta1.StandardInfraConfigSpecRouter
version **任意**	string

networkPolicy

名前	スキーマ
egress 任意	< io.k8s.api.networking.v1.NetworkPolicyEgressRule > array
ingress 任意	< io.k8s.api.networking.v1.NetworkPolicyIngressRule > array

D.1.3.14. io.enmasse.admin.v1beta1.StandardInfraConfigSpecAdmin

名前	スキーマ
podTemplate 任意	io.enmasse.admin.v1beta1.InfraConfigPodSpec
resources 任意	resources

resources

名前	スキーマ
memory 任意	string

D.1.3.15. io.enmasse.admin.v1beta1.StandardInfraConfigSpecBroker

名前	スキーマ
addressFullPolicy optional	enum (PAGE, BLOCK, FAIL)
connectorIdleTimeout optional	integer
connectorWorkerThreads optional	integer
podTemplate 任意	io.enmasse.admin.v1beta1.InfraConfigPodSpec
resources 任意	resources

名前	スキーマ
storageClassName **optional**	string
updatePersistentVolumeClaim **optional**	boolean

resources

名前	スキーマ
memory **任意**	string
storage **任意**	string

D.1.3.16. io.enmasse.admin.v1beta1.StandardInfraConfigSpecRouter

名前	スキーマ
idleTimeout **optional**	integer
initialHandshakeTimeout **optional**	integer
linkCapacity **任意**	integer
minAvailable **任意**	integer
minReplicas **optional**	integer
podTemplate **任意**	io.enmasse.admin.v1beta1.InfraConfigPodSpec
policy **任意**	policy
resources **任意**	resources

名前	スキーマ
workerThreads 任意	integer

policy

名前	スキーマ
maxConnections 任意	integer
maxConnectionsPerHost 任意	integer
maxConnectionsPerUser 任意	integer
maxReceiversPerConnection 任意	integer
maxSendersPerConnection 任意	integer
maxSessionsPerConnection 任意	integer

resources

名前	スキーマ
memory 任意	string

D.1.3.17. io.enmasse.admin.v1beta2.AddressPlan

名前	スキーマ
apiVersion 必須	enum (admin.enmasse.io/v1beta2)
kind 必須	列挙 (AddressPlan)
metadata 必須	ObjectMeta

名前	スキーマ
spec 必須	io.enmasse.admin.v1beta2.AddressPlanSpec

D.1.3.18. io.enmasse.admin.v1beta2.AddressPlanList

名前	スキーマ
apiVersion 必須	enum (admin.enmasse.io/v1beta2)
items 必須	< io.enmasse.admin.v1beta2.AddressPlan > array
kind 必須	列挙 (AddressPlanList)

D.1.3.19. io.enmasse.admin.v1beta2.AddressPlanSpec

名前	スキーマ
addressType 必須	string
displayName 必須	string
displayOrder 任意	integer
longDescription optional	string
partitions 任意	integer
resources 必須	resources
shortDescription 任意	string

resources

名前	スキーマ
broker **任意**	number
router **任意**	number

D.1.3.20. io.enmasse.admin.v1beta2.AddressSpacePlan

名前	スキーマ
apiVersion **必須**	enum (admin.enmasse.io/v1beta2)
kind **必須**	enum (AddressSpacePlan)
metadata **必須**	ObjectMeta
spec **必須**	io.enmasse.admin.v1beta2.AddressSpacePlanSpec

D.1.3.21. io.enmasse.admin.v1beta2.AddressSpacePlanList

名前	スキーマ
apiVersion **必須**	enum (admin.enmasse.io/v1beta2)
items **必須**	< io.enmasse.admin.v1beta2.AddressSpacePlan > array
kind **必須**	enum (AddressSpacePlanList)

D.1.3.22. io.enmasse.admin.v1beta2.AddressSpacePlanSpec

名前	スキーマ
addressPlans **必須**	< 文字列 > 配列
addressSpaceType **required**	string

名前	スキーマ
displayName 必須	string
displayOrder 任意	integer
infraConfigRef 必須	string
longDescription optional	string
resourceLimits required	resourceLimits
shortDescription 任意	string

resourceLimits

名前	スキーマ
aggregate 任意	number
broker 任意	number
router 任意	number

D.1.3.23. io.enmasse.user.v1beta1.MessagingUser

名前	スキーマ
apiVersion 必須	enum (user.enmasse.io/v1beta1)
kind 必須	列挙 (MessagingUser)
metadata 必須	ObjectMeta

名前	スキーマ
spec **必須**	io.enmasse.user.v1beta1.UserSpec

D.1.3.24. io.enmasse.user.v1beta1.MessagingUserList

名前	スキーマ
apiVersion **必須**	enum (user.enmasse.io/v1beta1)
items **必須**	< io.enmasse.user.v1beta1.MessagingUser > array
kind **必須**	enum (MessagingUserList)

D.1.3.25. io.enmasse.user.v1beta1.UserSpec

名前	スキーマ
authentication **任意**	authentication
authorization **任意**	< authorization > array
username **必須**	string

authentication

名前	説明	スキーマ
federatedUserid **任意**	'federated' タイプが指定されている場合にフェデレートするユーザーのユーザー ID。	string
federatedUsername **optional**	'federated' タイプが指定されている場合にフェデレートするユーザーのユーザー名。	string
パスワード **オプション**	'password' タイプが指定されている場合のパスワードの Base64 エンコード値。	string

名前	説明	スキーマ
provider **任意**	'federated' タイプが指定されている場合にフェデレーション ID に使用するプロバイダーの名前。	string
type **必須**		enum (password, serviceaccount)

認可

名前	スキーマ
addresses **任意**	< 文字列 > 配列
operations **必須**	< enum (send, recv, view, manage) > array

D.1.3.26. io.enmasse.v1beta1.Address

名前	スキーマ
apiVersion **必須**	enum (enmasse.io/v1beta1)
kind **必須**	enum (Address)
metadata **必須**	ObjectMeta
spec **必須**	io.enmasse.v1beta1.AddressSpec
status **任意**	io.enmasse.v1beta1.AddressStatus

D.1.3.27. io.enmasse.v1beta1.AddressList

名前	説明	スキーマ
apiVersion **必須**	Default : **"enmasse.io/v1beta1"**	enum (enmasse.io/v1beta1)

名前	説明	スキーマ
items **必須**		< io.enmasse.v1beta1.Address > array
kind **必須**		列挙 (AddressList)

D.1.3.28. io.enmasse.v1beta1.AddressSpace

名前	スキーマ
apiVersion **必須**	enum (enmasse.io/v1beta1)
kind **必須**	enum (AddressSpace)
metadata **必須**	ObjectMeta
spec **必須**	io.enmasse.v1beta1.AddressSpaceSpec
status **任意**	io.enmasse.v1beta1.AddressSpaceStatus

D.1.3.29. io.enmasse.v1beta1.AddressSpaceList

名前	説明	スキーマ
apiVersion **必須**	Default : **"enmasse.io/v1beta1"**	enum (enmasse.io/v1beta1)
items **必須**		< io.enmasse.v1beta1.AddressSpace > array
kind **必須**		enum (AddressSpaceList)

D.1.3.30. io.enmasse.v1beta1.AddressSpaceSpec

名前	説明	スキーマ
authenticationService **任意**		authenticationService
connectors **任意**	作成するコネクターの一覧表示。	< io.enmasse.v1beta1.AddressSpaceSpecConnector > array
endpoints **任意**		< endpoints > array
networkPolicy **任意**		networkPolicy
plan **必須**		string
type **必須**		io.enmasse.v1beta1.AddressSpaceType

authenticationService

名前	スキーマ
name **任意**	文字列
overrides **任意**	overrides
type **任意**	string

overrides

Name	スキーマ
host **任意**	string
port **任意**	integer

Name	スキーマ
realm **任意**	string

endpoints

名前	スキーマ
cert **任意**	cert
exports **任意**	< exports > array
expose **任意**	expose
name **任意**	string
service **任意**	string

cert

名前	スキーマ
provider **任意**	string
secretName **optional**	string
tlsCert **任意**	string
tlsKey **任意**	string

exports

名前	スキーマ
kind **任意**	enum (ConfigMap, Secret, Service)

名前	スキーマ
name **任意**	string

expose

名前	スキーマ
annotations **任意**	object
loadBalancerPorts **任意**	< 文字列 > 配列
loadBalancerSourceRanges **任意**	< 文字列 > 配列
routeHost **optional**	string
routeServicePort **optional**	string
routeTlsTermination **任意**	string
type **任意**	enum (route, loadbalancer)

networkPolicy

名前	スキーマ
egress **任意**	< io.k8s.api.networking.v1.NetworkPolicyEgressRule > array
ingress **任意**	< io.k8s.api.networking.v1.NetworkPolicyIngressRule > array

D.1.3.31. io.enmasse.v1beta1.AddressSpaceSpecConnector

名前	説明	スキーマ
addresses **任意**	このアドレス空間を介してアクセスできるようにするアドレス。	< addresses > array

名前	説明	スキーマ
credentials **任意**	エンドポイントへの接続時に使用される認証情報。'username' および 'password'、または 'secret' のいずれかを定義する必要があります。	credentials
endpointHosts **required**	接続するホストのリスト。少なくとも 1 つのエントリーが含まれている必要があります。	< endpointHosts > 配列
name **必須**	コネクターの名前。	string
tls **任意**	コネクターの TLS 設定。指定しない場合、TLS は使用されません。	tls

addresses

名前	説明	スキーマ
name **必須**	アドレスパターンの識別子。パターンを一意に識別するために使用	string
pattern **必須**	アドレスの照合に使用されるパターン。パターンには、コネクター名とスラッシュ ('myconnector/') が接頭辞として付けられます。パターンは、スラッシュ / で区切られた 1 つ以上のトークンで設定されます。トークンは、* 文字、# 文字、または /、*、# を含まない一連の文字のいずれかです。* トークンは、任意の 1 つのトークンに一致します。# トークンは、0 個以上のトークンに一致します。* は # よりも優先順位が高く、完全一致が最も優先順位が高くなります。	string

credentials

名前	説明	スキーマ
パスワード **オプション**	コネクターに使用するパスワード。'value' または 'secret' のいずれかを指定する必要があります。	password
username **任意**	コネクターに使用するユーザー名。'value' または 'secret' のいずれかを指定する必要があります。	username

password

名前	スキーマ
value **任意**	string
valueFromSecret **任意**	valueFromSecret

valueFromSecret

名前	説明	スキーマ
key **任意**	パスワードエントリーの検索に使用するキー。 **デフォルト: "password"**	string
name **任意**	パスワードを含むシークレットの名前。	string

username

名前	スキーマ
value **任意**	string
valueFromSecret **任意**	valueFromSecret

valueFromSecret

名前	説明	スキーマ
key **任意**	ユーザー名エントリーの検索に使用するキー。 **デフォルト: "username"**	string
name **任意**	ユーザー名を含むシークレットの名前。	string

endpointHosts

名前	説明	スキーマ
host **必須**	接続するホスト。	string

名前	説明	スキーマ
port **必須**	接続先の TCP ポート。	integer

tls

名前	説明	スキーマ
caCert **任意**	コネクターが使用する CA 証明書。'value' または 'secret' のいずれか。	cacert
clientCert **任意**	コネクターが使用するクライアント証明書。'value' または 'secret' のいずれか。	clientCert

cacert

名前	説明	スキーマ
value **任意**	CA 証明書の PEM エンコード値	string
valueFromSecret **任意**	コネクターが使用する CA 証明書を含むシークレット。	valueFromSecret

valueFromSecret

名前	説明	スキーマ
key **任意**	CA 証明書エントリーの検索に使用するキー。 **デフォルト : "ca.crt"**	string
name **任意**	CA 証明書を含むシークレットの名前。	string

clientCert

名前	説明	スキーマ
value **任意**	クライアント証明書の PEM エンコード値	string

名前	説明	スキーマ
valueFromSecret **任意**	コネクターが使用するクライアント証明書を含むシークレット。	valueFromSecret

valueFromSecret

名前	説明	スキーマ
key **任意**	クライアント証明書エントリーの検索に使用するキー。 **デフォルト** : **"ca.crt"**	string
name **任意**	クライアント証明書を含むシークレットの名前。	string

D.1.3.32. io.enmasse.v1beta1.AddressSpaceStatus

名前	説明	スキーマ
connectors **任意**	ステータス付きのコネクターの一覧表示。	< io.enmasse.v1beta1.AddressSpaceStatusConnector > array
endpointStatuses **optional**		< endpointStatuses > array
isReady **任意**		boolean
messages **任意**		< 文字列 > 配列

endpointStatuses

名前	スキーマ
cert **任意**	string
externalHost **任意**	string

名前	スキーマ
externalPorts **optional**	< externalPorts > 配列
name **任意**	string
serviceHost **optional**	string
servicePorts **optional**	< servicePorts > 配列

externalPorts

名前	スキーマ
name **任意**	string
port **任意**	integer

servicePorts

名前	スキーマ
name **任意**	string
port **任意**	integer

D.1.3.33. io.enmasse.v1beta1.AddressSpaceStatusConnector

名前	説明	スキーマ
isReady **任意**	コネクターが期待どおりに動作している場合は true、そうでない場合は false。	boolean
messages **任意**	コネクターの状態を説明するメッセージ。	< 文字列 > 配列
name **任意**	コネクターの名前。	string

D.1.3.34. io.enmasse.v1beta1.AddressSpaceType

AddressSpaceType は、アドレス空間のタイプ (standard、brokered) です。各タイプは、異なるタイプのアドレスとそれらのタイプのセマンティクスをサポートします。

Type : enum (standard, brokered)

D.1.3.35. io.enmasse.v1beta1.AddressSpec

名前	説明	スキーマ
address **必須**		string
forwarders **任意**	このアドレスに対して有効にするフォワーダーのリスト。	< io.enmasse.v1beta1.AddressSpecForwarder > array
plan **必須**		string
type **必須**		io.enmasse.v1beta1.AddressType

D.1.3.36. io.enmasse.v1beta1.AddressSpecForwarder

名前	説明	スキーマ
direction **必須**	フォワーダーの方向。in は、remoteAddress からこのアドレスにプルすることを意味します。out は、このアドレスから remoteAddress にプッシュすることを意味します。	enum (in, out)
name **必須**	フォワーダーの名前。	string
remoteAddress **required**	メッセージを送受信するリモートアドレス。	string

D.1.3.37. io.enmasse.v1beta1.AddressStatus

名前	説明	スキーマ
forwarders **任意**	ステータス付きのフォワーダーのリスト。	< io.enmasse.v1beta1.AddressStatusForwarder > array

名前	説明	スキーマ
isReady **任意**		boolean
messages **任意**		< 文字列 > 配列
phase **任意**		enum (Pending, Configuring, Active, Failed, Terminating)

D.1.3.38. io.enmasse.v1beta1.AddressStatusForwarder

名前	説明	スキーマ
isReady **任意**	フォワーダーが期待どおりに動作している場合は true、そうでない場合は false。	boolean
messages **任意**	フォワーダーの状態を説明するメッセージ。	< 文字列 > 配列
name **任意**	フォワーダーの名前。	string

D.1.3.39. io.enmasse.v1beta1.AddressType

アドレスのタイプ (queue、topic など)。各アドレスタイプは、さまざまな種類のメッセージングセマンティクスをサポートします。

タイプ: enum (queue, topic, anycast, multicast)

D.1.3.40. io.k8s.api.networking.v1.IPBlock

IPBlock は、特定の CIDR を記述します (例:"192.168.1.1/24") は、NetworkPolicySpec の podSelector によって一致する Pod に許可されます。例外エントリーは、このルールに含めるべきではない CIDR を記述します。

名前	説明	スキーマ
cidr **必須**	CIDR は、IP ブロックを表す文字列です。有効な例は 192.168.1.1/24 です。	string
except **任意**	Except は、IP ブロックに含めるべきではない CIDR のスライスです。有効な例は 192.168.1.1/24 です。ただし、CIDR の範囲外の場合は、値が拒否されます。	< 文字列 > 配列

D.1.3.41. io.k8s.api.networking.v1.NetworkPolicyEgressRule

NetworkPolicyEgressRule は、NetworkPolicySpec の podSelector によって一致する Pod から許可される特定のトラフィックセットを記述します。トラフィックは、ポートと宛先の両方に一致する必要があります。このタイプは 1.8 のベータレベルです

名前	説明	スキーマ
ports **任意**	発信トラフィックの宛先ポートのリスト。このリストの各項目は、論理 OR を使用して結合されます。このフィールドが空または欠落している場合、このルールはすべてのポートに一致します (トラフィックはポートによって制限されません)。このフィールドが存在し、少なくとも 1 つのアイテムが含まれている場合、このルールは、トラフィックがリスト内の少なくとも 1 つのポートに一致する場合にのみトラフィックを許可します。	< io.k8s.api.networking.v1.NetworkPolicyPort > 配列
to **任意**	このルール用に選択された Pod の送信トラフィックの宛先のリスト。このリストの項目は、論理演算子 OR を使用して組み合わせます。このフィールドが空または欠落している場合、このルールはすべての宛先に一致します (トラフィックは宛先によって制限されません)。このフィールドが存在し、少なくとも 1 つのアイテムが含まれている場合、このルールは、トラフィックが to リストの少なくとも 1 つのアイテムと一致する場合にのみトラフィックを許可します。	< io.k8s.api.networking.v1.NetworkPolicyPeer > 配列

D.1.3.42. io.k8s.api.networking.v1.NetworkPolicyIngressRule

NetworkPolicyIngressRule は、NetworkPolicySpec の podSelector によって一致する Pod に許可される特定のトラフィックセットを記述します。トラフィックは、ポートと送信元の両方に一致する必要があります。

名前	説明	スキーマ
from **任意**	このルール用に選択された Pod にアクセスできる必要があるソースのリスト。このリストの項目は、論理演算子 OR を使用して組み合わせます。このフィールドが空または欠落している場合、このルールはすべてのソースに一致します (トラフィックはソースによって制限されていません)。このフィールドが存在し、少なくとも 1 つのアイテムが含まれている場合、このルールは、トラフィックが from リストの少なくとも 1 つのアイテムと一致する場合にのみトラフィックを許可します。	< io.k8s.api.networking.v1.NetworkPolicyPeer > 配列
ports **任意**	このルール用に選択された Pod でアクセス可能にする必要があるポートのリスト。このリストの各項目は、論理 OR を使用して結合されます。このフィールドが空または欠落している場合、このルールはすべてのポートに一致します (トラフィックはポートによって制限されません)。このフィールドが存在し、少なくとも 1 つのアイテムが含まれている場合、このルールは、トラフィックがリスト内の少なくとも 1 つのポートに一致する場合にのみトラフィックを許可します。	< io.k8s.api.networking.v1.NetworkPolicyPort > 配列

D.1.3.43. io.k8s.api.networking.v1.NetworkPolicyPeer

NetworkPolicyPeer は、受信トラフィックを許可するピアについて記述します。フィールドの特定の組み合わせのみが許可されます

名前	説明	スキーマ
ipBlock **任意**	IPBlock は、特定の IPBlock に関するポリシーを定義します。このフィールドが設定されている場合は、他のフィールドを設定することはできません。	io.k8s.api.networking.v1.IPBlock
namespaceSelector **任意**	クラススコープのラベルを使用して名前空間を選択します。このフィールドは、標準のラベルセレクターセマンティクスに従います。存在するが空の場合、すべての名前空間が選択されます。 PodSelector も設定されている場合、NetworkPolicyPeer は全体として、NamespaceSelector によって選択された名前空間で PodSelector に一致する Pod を選択します。それ以外の場合は、NamespaceSelector によって選択されたネームスペース内のすべての Pod を選択します。	io.k8s.apimachinery.pkg.apis.meta.v1.LabelSelector
podSelector **optional**	Pod を選択するラベルセレクターです。このフィールドは、標準のラベルセレクターセマンティクスに従います。存在するが空の場合、すべての Pod が選択されます。 NamespaceSelector も設定されている場合、NetworkPolicyPeer は全体として、NamespaceSelector によって選択された名前空間で PodSelector に一致する Pod を選択します。それ以外の場合は、ポリシー自体の名前空間で PodSelector に一致する Pod を選択します。	io.k8s.apimachinery.pkg.apis.meta.v1.LabelSelector

D.1.3.44. io.k8s.api.networking.v1.NetworkPolicyPort

NetworkPolicyPort は、トラフィックを許可するポートを記述します

名前	説明	スキーマ
port **任意**	指定されたプロトコルのポート。これは、Pod の数値ポートまたは名前付きポートのいずれかです。このフィールドが指定されていない場合、これはすべてのポート名と番号に一致します。	io.k8s.apimachinery.pkg.util.intstr.IntOrString
protocol **任意**	トラフィックが一致する必要があるプロトコル (TCP または UDP)。指定しない場合、このフィールドのデフォルトは TCP です。	string

D.1.3.45. io.k8s.apimachinery.pkg.apis.meta.v1.LabelSelector

ラベルセレクターとは、一連のリソースに対するラベルクエリー機能です。matchLabels と matchExpressions の結果は AND を使用して結合されます。ラベルセレクターが空の場合は、全オブジェクトをマッチします。ラベルセレクターが null の場合は、どのオブジェクトもマッチしません。

名前	説明	スキーマ
matchExpressions **任意**	matchExpressions はラベルセレクターの要件の一覧です。要件は AND で結合されます。	< io.k8s.apimachinery.pkg.apis.meta.v1.LabelSelectorRequirement > array
matchLabels **任意**	matchLabels は、{key,value} ペアのマップです。MatchLabels マップの 1 つの {key,value} は matchExpressions の要素と同じで、キーフィールドには key、演算子には In、値配列には value のみが含まれます。要件は AND で結合されます。	< string, string > マップ

D.1.3.46. io.k8s.apimachinery.pkg.apis.meta.v1.LabelSelectorRequirement

ラベルセレクター要件は、値、キー、およびキーと値を関連付ける Operator を含むセレクターです。

名前	説明	スキーマ
key **必須**	key は、セレクターの適用先のラベルキーです。	文字列
operator **必須**	operator はキーと値のセットの関係を表します。有効な演算子は In、NotIn、Exists、および DoesNotExist です。	文字列
values **任意**	values は文字列値の配列です。operator が In または NotIn の場合には、values 配列を空白にできません。operator が Exists または DoesNotExist の場合には、values 配列は空白でなければなりません。この配列は、ストラテジーに基づいたマージパッチの適用中に置き換えられます。	< 文字列 > 配列

D.1.3.47. io.k8s.apimachinery.pkg.util.intstr.IntOrString

IntOrString は int32 または文字列を保持することができるタイプです。JSON または YAML のマーシャリングおよびアンマーシャリングで使用すると、内部タイプを生成または消費します。これにより、たとえば名前または数字を許可する JSON フィールドなどを設定できます。

タイプ: 文字列 (整数または文字列)

付録E サブスクリプションの使用

AMQ Online は、ソフトウェアサブスクリプションを通じて提供されます。サブスクリプションを管理するには、Red Hat カスタマーポータルでアカウントにアクセスします。

アカウントへのアクセス

1. access.redhat.com に移動します。
2. アカウントがない場合は、作成します。
3. アカウントにログインします。

サブスクリプションのアクティベート

1. access.redhat.com に移動します。
2. **サブスクリプション** に移動します。
3. **Activate a subscription** に移動し、16 桁のアクティベーション番号を入力します。

zip および tar ファイルのダウンロード

zip または tar ファイルにアクセスするには、Red Hat カスタマーポータルを使用して、ダウンロードする関連ファイルを検索します。RPM パッケージを使用している場合は、この手順は必要ありません。

1. ブラウザーを開き、access.redhat.com/downloads で Red Hat カスタマーポータルの **Product Downloads** ページにログインします。
2. **JBOSS INTEGRATION AND AUTOMATION** カテゴリーの **Red Hat AMQ Online** エントリーを見つけます。
3. 目的の AMQ Online 製品を選択します。Software Downloads ページが開きます。
4. コンポーネントの **ダウンロード** リンクをクリックします。

パッケージ用のシステムの登録

RPM パッケージを Red Hat Enterprise Linux にインストールするには、システムが登録されている必要があります。zip または tar ファイルを使用している場合、この手順は必要ありません。

1. access.redhat.com に移動します。
2. **Registration Assistant** に移動します。
3. ご使用の OS バージョンを選択し、次のページに進みます。
4. システムターミナルで listed コマンドを使用して、登録を完了します。

詳細は、How to Register and Subscribe a System to the Red Hat Customer Portal を参照してください。

改訂日時：2023-01-28 11:54:15 +1000