デベロッパーガイド

Amazon MQ

Amazon MQ: デベロッパーガイド

Table of Contents

Amazon MQ とは

Amazon MQ は、メッセージブローカーのセットアップ、運用、保守を管理する、Apache ActiveMQ Classic および RabbitMQ 向けのマネージドメッセージブローカーサービスです。業界標準のメッセージングプロトコルを使用して新しい Amazon MQ ブローカーを作成することも、既存のメッセージブローカーからメッセージングコードを書き換えずに Amazon MQ に移行することもできます。

ブローカーは、Amazon MQ で実行されるメッセージブローカー環境です。これは、Amazon MQ の基本的な構成要素です。メッセージブローカーを使用すると、ソフトウェアアプリケーションおよびコンポーネントが、さまざまなプログラミング言語、オペレーティングシステム、正式なメッセージングプロトコルを使用して通信できます。Amazon MQ ブローカーは、大規模なクラウドネイティブアプリケーションとコンポーネント間の通信に使用できます。

トピック

- Amazon MQ の機能
- Amazon MQ の使用を開始するにはどうすればよいですか。
- Amazon MQ にフィードバックを提供するにはどうすればよいですか。

Amazon MQ の機能

マネージドメンテナンスとバージョンアップグレード

Amazon MQ は、スケジュールされたメンテナンスウィンドウ内で、メッセージブローカーのメンテナンスとバージョンアップグレードを実行します。

CloudWatch によるブローカーのモニタリング

Amazon MQ は Amazon CloudWatch と統合されているため、ブローカーとキューのメトリクスを表示および分析できます。メトリクスの表示と分析は、Amazon MQ コンソール、CloudWatch コンソール、コマンドライン、および API から行うことができます。メトリクスは自動的に収集され、1 分おきに CloudWatch にプッシュされます。

セキュリティ

Amazon MQ は、保管中および転送中のメッセージの暗号化を提供します。ブローカーへの接続には SSL が使用され、アクセスは Amazon VPC 内のプライベートエンドポイントに制限できます。さ

らに、AWS Identity and Access Management (IAM) を使用して、IAM ユーザーとグループが特定の Amazon MQ ブローカーに対して実行できるアクションを制御することも可能です。

Amazon MQ での RabbitMQ のクォーラムキュー

クォーラムキューは、1 つのリーダーノード (プライマリレプリカ) と複数のフォロワーノード (その他のレプリカ) で構成されるレプリケーション型のキュータイプです。各ノードは別々のアベイラビリティーゾーンにあるため、1 つのノードが一時的に利用できなくなっても、メッセージ配信は別のアベイラビリティーゾーンにある新しく選出されたリーダーレプリカによって続行されます。クォーラムキューは、メッセージが失敗し、キューに何度も入れ直されたときに発生する有害メッセージを処理するために役立ちます。

Amazon MQ for ActiveMQ のクロスリージョンデータレプリケーション

クロスリージョンデータレプリケーション (CRDR) 機能は、プライマリ AWS リージョンにあるプライマリブローカーからレプリカリージョンにあるレプリカブローカーへの非同期メッセージレプリケーションを可能にします。Amazon MQ API にフェイルオーバーリクエストを発行すると、現在のレプリカブローカーはプライマリブローカーのロールに昇格され、現在のプライマリブローカーはレプリカのロールに降格されます。

Amazon MQ の使用を開始するにはどうすればよいですか。

Amazon MQ で ActiveMQ の使用を開始するには、次のドキュメントを参照してください。

- 開始方法: ActiveMQ ブローカーの作成と接続
- the section called “ブローカーのデプロイ”
- ActiveMQ チュートリアル
- the section called “Amazon MQ for ActiveMQ のベストプラクティス”

Amazon MQ で RabbitMQ の使用を開始するには、次のドキュメントを参照してください。

- 開始方法: RabbitMQ ブローカーの作成と接続
- the section called “RabbitMQ ブローカーのデプロイ”
- the section called “RabbitMQ のチュートリアル”
- the section called “Amazon MQ for RabbitMQ のベストプラクティス”

Amazon MQ REST API については、Amazon MQ REST API リファレンスを参照してください。

Amazon MQ AWS CLI コマンドについては、AWS CLI コマンドリファレンスで Amazon MQ を参照してください。

Amazon MQ にフィードバックを提供するにはどうすればよいですか。

ドキュメントに関するフィードバックをお待ちしております。右側にある高評価アイコンと低評価アイコンを使用してフィードバックを送信するか、下にリンクされている「フィードバックを送信」フォームを使用できます。

Amazon MQ チームに連絡するには、Amazon MQ ディスカッションフォーラムを使用してください。

Amazon MQ のセットアップ

Amazon MQ を使用する前に、以下のステップを完了しておく必要があります。

トピック

ステップ 1: 前提条件

にサインアップする AWS アカウント

がない場合は AWS アカウント、次の手順を実行して作成します。

にサインアップするには AWS アカウント

1. https://portal.aws.amazon.com/billing/signup を開きます。
2. オンラインの手順に従います。

   サインアップ手順の一環として、電話またはテキストメッセージを受け取り、電話キーパッドで検証コードを入力します。

   にサインアップすると AWS アカウント、 AWS アカウントのルートユーザー が作成されます。ルートユーザーには、アカウントのすべての AWS のサービス とリソースへのアクセス権があります。セキュリティベストプラクティスとして、ユーザーに管理アクセス権を割り当て、ルートユーザーアクセスが必要なタスクの実行にはルートユーザーのみを使用するようにしてください。

AWS サインアッププロセスが完了すると、 から確認メールが送信されます。https://aws.amazon.com/ の [マイアカウント] をクリックして、いつでもアカウントの現在のアクティビティを表示し、アカウントを管理することができます。

管理アクセスを持つユーザーを作成する

にサインアップしたら AWS アカウント、日常的なタスクにルートユーザーを使用しないように AWS アカウントのルートユーザー、 を保護し AWS IAM アイデンティティセンター、 を有効にして管理ユーザーを作成します。

を保護する AWS アカウントのルートユーザー

1. ルートユーザーを選択し、 AWS アカウント E メールアドレスを入力して、アカウント所有者AWS マネジメントコンソールとして にサインインします。次のページでパスワードを入力します。

   ルートユーザーを使用してサインインする方法については、「AWS サインイン ユーザーガイド」の「ルートユーザーとしてサインインする」を参照してください。
2. ルートユーザーの多要素認証 (MFA) を有効にします。

   手順については、IAM ユーザーガイドの AWS アカウント 「ルートユーザー (コンソール) の仮想 MFA デバイスを有効にする」を参照してください。

管理アクセスを持つユーザーを作成する

1. IAM アイデンティティセンターを有効にします。

   手順については、「AWS IAM アイデンティティセンター ユーザーガイド」の「AWS IAM アイデンティティセンターの有効化」を参照してください。
2. IAM アイデンティティセンターで、ユーザーに管理アクセスを付与します。

   を ID ソース IAM アイデンティティセンターディレクトリ として使用する方法のチュートリアルについては、「 AWS IAM アイデンティティセンター ユーザーガイド」の「デフォルトを使用してユーザーアクセスを設定する IAM アイデンティティセンターディレクトリ」を参照してください。

管理アクセス権を持つユーザーとしてサインインする

- IAM アイデンティティセンターのユーザーとしてサインインするには、IAM アイデンティティセンターのユーザーの作成時に E メールアドレスに送信されたサインイン URL を使用します。

  IAM Identity Center ユーザーを使用してサインインする方法については、AWS サインイン「 ユーザーガイド」の AWS 「 アクセスポータルにサインインする」を参照してください。

追加のユーザーにアクセス権を割り当てる

1. IAM アイデンティティセンターで、最小特権のアクセス許可を適用するというベストプラクティスに従ったアクセス許可セットを作成します。

   手順については、「AWS IAM アイデンティティセンター ユーザーガイド」の「アクセス許可セットを作成する」を参照してください。

2. グループにユーザーを割り当て、そのグループにシングルサインオンアクセス権を割り当てます。

   手順については、「AWS IAM アイデンティティセンター ユーザーガイド」の「グループを追加する」を参照してください。

ステップ 2: ユーザーを作成して AWS 認証情報を取得する

ユーザーが の AWS 外部とやり取りする場合は、プログラムによるアクセスが必要です AWS マネジメントコンソール。プログラムによるアクセスを許可する方法は、 がアクセスするユーザーのタイプによって異なります AWS。

ユーザーにプログラムによるアクセス権を付与するには、以下のいずれかのオプションを選択します。

プログラムによるアクセス権を必要とするユーザー	目的	方法
IAM	(推奨) コンソール認証情報を一時的な認証情報として使用して AWS CLI、、 AWS SDKs、または AWS APIs。	使用するインターフェイスの指示に従ってください。 • については AWS CLI、AWS Command Line Interface 「 ユーザーガイド」のAWS「ローカル開発用のログイン」を参照してください。 • AWS SDKs「 SDK およびツールリファレンスガイド」の「Login for AWS

プログラムによるアクセス権を必要とするユーザー	目的	方法
		local development」を参照してください。 AWS SDKs
ワークフォースアイデンティティ (IAM アイデンティティセンターで管理されているユーザー)	一時的な認証情報を使用して AWS CLI、、 AWS SDKs、または AWS APIs。	使用するインターフェイスの指示に従ってください。 • については AWS CLI、「 AWS Command Line Interface ユーザーガイド」の「 を使用する AWS CLI ように AWS IAM アイデンティティセンターを設定する」を参照してください。 • AWS SDKs、ツール、API については、 AWS APIs「 SDK およびツールリファレンスガイド」の「IAM アイデンティティセンター認証」を参照してください。 AWS SDKs
IAM	一時的な認証情報を使用して AWS CLI、、 AWS SDKs、または AWS APIs。	「IAM ユーザーガイド」の「 AWS リソースでの一時的な認証情報の使用」の手順に従います。

プログラムによるアクセス権を必要とするユーザー	目的	方法
IAM	(非推奨) 長期認証情報を使用して、AWS CLI、 AWS SDKs、または AWS APIs。	使用するインターフェイスの指示に従ってください。 • については AWS CLI、「 AWS Command Line Interface ユーザーガイド」の「IAM ユーザー認証情報を使用した認証」を参照してください。 • AWS SDKs「 SDK とツールリファレンスガイド」の「長期認証情報を使用した認証」を参照してください。 AWS SDKs • API AWS APIs「IAM ユーザーガイド」の「IAM ユーザーのアクセスキーの管理」を参照してください。

ステップ 3: サンプルコードの使用準備を整える

以下のチュートリアルでは、 を使用して Amazon MQ ブローカーを操作する方法と AWS マネジメントコンソール 、Amazon MQ for ActiveMQ および Amazon MQ for RabbitMQ ブローカーにプログラムで接続する方法を示します。ActiveMQ Java サンプルコードを使用するには、Java Standard Edition Development Kit をインストールして、コードにいくつかの変更を行う必要があります。

Amazon MQ REST API と AWS SDKs を使用して、プログラムでブローカーを作成および管理することもできます。

次の手順

Amazon MQ を使用する準備ができたので、ブローカーを作成することによって使用を開始します。ブローカーのエンジンタイプに応じて、Amazon MQ for ActiveMQ ブローカーに Java アプ

リケーションを接続するか、RabbitMQ Java クライアントライブラリを使用して Amazon MQ for RabbitMQ ブローカーに JVM ベースのアプリケーションを接続します。

開始方法: ActiveMQ ブローカーの作成と接続

ブローカーは、Amazon MQ で実行されるメッセージブローカー環境です。これは、Amazon MQ の基本的な構成要素です。ブローカーインスタンスのクラス (`m5`) およびサイズ (`large`、`medium`) を組み合わせた説明は、ブローカーインスタンスタイプ (`mq.m5.large` など) と呼ばれます。詳細については、「Amazon MQ for ActiveMQ ブローカーとは」を参照してください。

ActiveMQ ブローカーの作成

最初に実行する最も一般的な Amazon MQ タスクは、ブローカーの作成です。以下の例では、AWS マネジメントコンソールを使用して基本的なブローカーを作成する方法を説明します。

1. Amazon MQ コンソールにサインインします。
2. [Select broker engine] (ブローカーエンジンの選択) ページで [Apache ActiveMQ] を選択します。
3. [Select deployment and storage] (デプロイとストレージタイプの選択) ページの [Deployment mode and storage type] (デプロイモードとストレージタイプ) セクションで、以下を実行します。

   a. [Deployment mode] (デプロイモード) を選択します ([Active/standby broker] (アクティブ/スタンバイブローカー)など)。詳細については、「Amazon MQ for ActiveMQ ブローカーのデプロイオプション」を参照してください。

      - 単一インスタンスブローカーは 1 つのアベイラビリティーゾーンにある 1 つのブローカーで構成されます。ブローカーは、アプリケーション、および Amazon EBS または Amazon EFS ストレージボリュームと通信します。詳細については、「オプション 1: Amazon MQ 単一インスタンスブローカー」を参照してください。
      - 高可用性対応のアクティブ/スタンバイブローカーは、2 つの異なるアベイラビリティーゾーンにある 2 つのブローカーで構成され、冗長ペアで設定されます。これらのブローカーは、アプリケーションおよび Amazon EFS と同期的に通信します。詳細については、「オプション 2: 高可用性対応の Amazon MQ アクティブ/スタンバイブローカー」を参照してください。

   b. [Storage type] (ストレージタイプ) を選択します (EBS など)。詳細については、「Storage」を参照してください。

> **Note**
>
> Amazon EBS は単一のアベイラビリティーゾーン内でデータをレプリケートし、ActiveMQ アクティブ/スタンバイデプロイモードをサポートしません。

c. [Next] (次へ) をクリックします。

4. [Configure settings] (設定の定義) ページの [Details] (詳細) セクションで、以下を実行します。

a. [Broker name] (ブローカー名) を入力します。

> **Important**
>
> 個人を特定できる情報 (PII) などの機密情報や秘匿性の高い情報はタグに追加しないでください。ブローカー名は、CloudWatch Logs を含む他の AWS サービスからアクセスできます。ブローカー名は、プライベートデータや機密データとして使用することを意図していません。

> **Note**
>
> [追加の設定] セクションでは、以下を設定することもできます。
>
> - 設定
> - CloudWatch ログ
> - プライベートアクセス
> - ブローカーメンテナンスウィンドウ

b. [Broker instance type] (ブローカーインスタンスタイプ) を選択します (mq.m5.large など)。詳細については、「Broker instance types」を参照してください。

5. [ActiveMQ Web Console access] (ActiveMQ ウェブコンソールアクセス) セクションで、[Username] (ユーザーネーム) と [Password] (パスワード) を入力します。ブローカーのユーザー名とパスワードには、以下の制限が適用されます:

- ユーザーネームに使用できるのは、英数字、ダッシュ、ピリオド、アンダースコア、およびチルダ (- . _ ~) のみです。

- パスワードは 12 文字以上の長さで、一意の文字を少なくとも 4 つ含める必要があり、カンマ、コロン、または等号 (,:=) は使用できません。

> **Important**
>
> 個人を特定できる情報 (PII) などの機密情報や秘匿性の高い情報はブローカーのユーザー名に追加しないでください。ブローカーのユーザー名は、CloudWatch Logs を含む他の AWS サービスからアクセスできます。ブローカーのユーザー名は、プライベートデータや機密データとして使用することを意図していません。

6. [Deploy] (デプロイ) をクリックします。

   Amazon MQ がブローカーを作成している間は、[Creation in progress] (作成中) ステータスが表示されます。

   ブローカーの作成には約 15 分かかります。

   ブローカーが正常に作成されると、Amazon MQ が [Running] (実行中) ステータスを表示します。

7. [***MyBroker***] を選択します。

   [MyBroker] ページの [接続] セクションにあるブローカーの [ActiveMQ web console] の URL をメモしておきます。以下はその例です。

   ```
   https://b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-east-2.amazonaws.com:8162
   ```

   また、ブローカーのワイヤレベルプロトコルの [Endpoints] (エンドポイント) もメモしておきます。以下は、OpenWire エンドポイントの例です。

   ```
   ssl://b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-east-2.amazonaws.com:61617
   ```

開始方法: RabbitMQ ブローカーの作成と接続

ブローカーは、Amazon MQ で実行されるメッセージブローカー環境です。これは、Amazon MQ の基本的な構成要素です。ブローカーインスタンスのクラス (`m5`) およびサイズ (`large`、`medium`) を組み合わせた説明は、ブローカーインスタンスタイプ (`mq.m5.large` など) と呼ばれます。詳細については、「Amazon MQ for RabbitMQ ブローカーとは」を参照してください。

RabbitMQ ブローカーを作成する

最初に実行する最も一般的な Amazon MQ タスクは、ブローカーの作成です。以下の例では、AWS マネジメントコンソールを使用して基本的なブローカーを作成する方法を説明します。

Amazon MQ for RabbitMQ ブローカーを作成するときは、RabbitMQ のブローカー設定のベストプラクティスに従って、ブローカーのパフォーマンスを最大化し、メッセージのスループット効率を最適化します。

1. Amazon MQ コンソールにサインインします。
2. [Select broker engine] (ブローカーエンジンの選択) ページで [RabbitMQ] を選択し、[Next] (次へ) をクリックします。
3. [Select deployment mode] (デプロイモードの選択) ページで [Deployment mode] (デプロイモード) ([Cluster deployment] (クラスターのデプロイ) など) を選択して、[Next] (次へ) をクリックします。

   - 単一インスタンスブローカーは、Network Load Balancer (NLB) の内側にある 1 つのアベイラビリティーゾーン内の 1 つのブローカーで構成されます。ブローカーは、アプリケーション、および Amazon EBS ストレージボリュームと通信します。詳細については、「オプション 1: Amazon MQ for RabbitMQ 単一インスタンスブローカー」を参照してください。
   - 高可用性対応の RabbitMQ クラスターデプロイは、Network Load Balancer の内側にある 3 つの RabbitMQ ブローカーノードの論理グループで、それぞれがユーザー、キュー、および複数のアベイラビリティーゾーン (AZ) 間の分散状態を共有します。詳細については、「オプション 2: Amazon MQ for RabbitMQ クラスターデプロイ」を参照してください。
4. [Configure settings] (設定の定義) ページの [Details] (詳細) セクションで、以下を実行します。

   a. [Broker name] (ブローカー名) を入力します。

> **Important**
>
> 個人を特定できる情報 (PII) などの機密情報や秘匿性の高い情報はタグに追加しないでください。ブローカー名は、CloudWatch Logs を含む他の AWS サービスからアクセスできます。ブローカー名は、プライベートデータや機密データとして使用することを意図していません。

b. [ブローカーインスタンスタイプ] を選択します ([mq.m7g.large] など)。詳細については、「Broker instance types」を参照してください。

5. [Configure settings] (設定の定義) ページの [RabbitMQ access] (RabbitMQ アクセス) セクションで、[Username] (ユーザーネーム) と [Password] (パスワード) を入力します。ブローカーのサインイン認証情報には以下の制限が適用されます。

   - ユーザーネームに使用できるのは、英数字、ダッシュ、ピリオド、およびアンダースコア (- . _) のみです。この値にチルダ (~) 文字を含めることはできません。Amazon MQ では、ユーザーネームとしての `guest` の使用が禁止されています。
   - パスワードは 12 文字以上の長さで、一意の文字を少なくとも 4 つ含める必要があり、カンマ、コロン、または等号 (,:=) は使用できません。

> **Important**
>
> 個人を特定できる情報 (PII) などの機密情報や秘匿性の高い情報はブローカーのユーザー名に追加しないでください。ブローカーのユーザー名は、CloudWatch Logs を含む他の AWS サービスからアクセスできます。ブローカーのユーザー名は、プライベートデータや機密データとして使用することを意図していません。

> **Note**
>
> [追加の設定] セクションでは、以下を設定することもできます。
>
> - 設定
> - CloudWatch ログ
> - プライベートアクセス
> - ブローカーメンテナンスウィンドウ

6. [Next] (次へ) をクリックします。
7. [Review and create] (確認と作成) ページで、選択内容を確認し、必要に応じて編集することができます。
8. [Create broker] (ブローカーの作成) をクリックします。

   Amazon MQ がブローカーを作成している間は、[Creation in progress] (作成中) ステータスが表示されます。

   ブローカーの作成には約 15 分かかります。

   ブローカーが正常に作成されると、Amazon MQ が [Running] (実行中) ステータスを表示します。
9. [***MyBroker***] を選択します。

   [MyBroker] ページの [接続] セクションにあるブローカーの [RabbitMQ web console] の URL をメモしておきます。以下はその例です。

   ```
   https://b-c8349341-ec91-4a78-ad9c-a57f23f235bb.mq.us-west-2.on.aws
   ```

   ブローカーの secure-AMQP エンドポイントもメモしておきます。以下は、リスナーポート 5671 を公開する amqps エンドポイントの例です。

   ```
   amqps://b-c8349341-ec91-4a78-ad9c-a57f23f235bb.mq.us-west-2.on.aws:5671
   ```

Amazon MQ ブローカーの管理

ブローカーを作成したら、Amazon MQ ブローカーのさまざまなコンポーネントを管理して維持できます。

トピック

Amazon MQ への接続

サービスエンドポイントとブローカーエンドポイントを使用して、他の AWS のサービスから Amazon MQ に接続できます。

サービスエンドポイント

Amazon MQ サービス API には、次の接続方法が使用されます。

ドメイン	接続方法
`mq.region.amazonaws.com`	IPv4
`mq.region.api.aws`	デュアルスタック (IPv4 および IPv6)
`mq-fips.region.amazonaws.com`	IPv4 のみを使用する FIPS

ドメイン	接続方法
mq-fips.*region*.api.aws	デュアルスタックを使用する FIPS

ブローカーエンドポイント

Amazon MQ ブローカーには、次の接続方法が使用されます。

ドメイン	接続方法
brokerId.mq.*region*.amazonaws.com	IPv4
brokerId.mq.*region*.on.aws	デュアルスタック (IPv4 および IPv6) **Note** Amazon MQ for ActiveMQ ブローカーはデュアルスタックをサポートしていません。

デュアルスタック (IPv4 および IPv6) エンドポイントを使用して Amazon MQ に接続する

デュアルスタックエンドポイントは、IPv4 と IPv6 トラフィックの両方をサポートします。デュアルスタックのエンドポイントにリクエストを行うと、エンドポイント URL は IPv4 または IPv6 アドレスに解決されます。デュアルスタックと FIPS エンドポイント詳細については、「SDK リファレンスガイド」を参照してください。

Amazon MQ はリージョンのデュアルスタックエンドポイントをサポートしており、エンドポイント名の一部として AWS リージョンを指定する必要があります。デュアルスタックエンドポイント名には、次の命名規則が使用されます。mq.*region*.api.aws例えば、eu-west-1 リージョンのデュアルスタックエンドポイント名は、mq.eu-west-1.api.aws です。

Amazon MQ エンドポイントの完全なリストについては、「AWS 全般のリファレンス」を参照してください。

AWS PrivateLink を使用して Amazon MQ に接続する

IPv4 および IPv6 をサポートする Amazon MQ API 用の AWS PrivateLink エンドポイントは、トラフィックをパブリックインターネットに公開することなく、仮想プライベートクラウド (VPC) と Amazon MQ API 間のプライベート接続を提供します。

> **Note**
>
> PrivateLink のサポートは、ブローカーエンドポイントではなく、Amazon MQ API エンドポイントでのみ使用できます。ブローカーエンドポイントへのプライベート接続の詳細については、「Configuring a private Amazon MQ broker」を参照してください。

PrivateLink を使用して Amazon MQ API にアクセスするには、まず接続元の特定の VPC にインターフェイス VPC エンドポイントを作成する必要があります。VPC エンドポイントを作成するときは、FIPS エンドポイントにサービス名 `com.amazonaws.region.mq-fips` または `com.amazonaws.region.mq` を使用します。

AWS CLI または SDK を使用して Amazon MQ を呼び出す場合は、デュアルスタックドメイン名 `mq.region.api.aws`または `mq-fips.region.api.aws` を使用するエンドポイント URL を指定する必要があります。PrivateLink for Amazon MQ は、`amazonaws.com` で終わるデフォルトのドメイン名をサポートしていません。詳細については、「SDK リファレンスガイド」の「デュアルスタックおよび FIPS エンドポイント」を参照してください。

次の CLI の例は、Amazon MQ VPC エンドポイントを介してアジアパシフィック (シドニー) リージョンで `describe-broker-engine-type` を呼び出す方法を示しています。

```
AWS_USE_DUALSTACK=true aws mq describe-broker-engine-types --region ap-southeast-2
```

CLI でエンドポイントを設定するその他の方法については、「AWS CLI でのエンドポイントの使用」を参照してください。

VPC エンドポイントポリシーを使用して、VPC エンドポイントへのユーザーアクセスを決定することもできます。詳細については、「エンドポイントポリシーを使用して VPC エンドポイントへのアクセスを制御する」を参照してください。

Amazon MQ ブローカーの認証と認可

Amazon MQ には、組織の要件に従ってメッセージングインフラストラクチャを保護するための複数の認証および認可方法が用意されています。

Amazon MQ for RabbitMQ の認証と認可

Amazon MQ for RabbitMQ は、次の認証および認可方法をサポートしています。

シンプルな認証と認可

この方法では、ブローカーユーザーは内部的に RabbitMQ ブローカーに保存され、ウェブコンソールまたは管理 API を介して管理されます。vhost、交換、キュー、トピックのアクセス許可は、RabbitMQ で直接設定されます。これがデフォルトの方法です。詳細については、「シンプルな認証と認可」を参照してください。

OAuth 2.0 の認証と認可

この方法では、ブローカーユーザーとそのアクセス許可は、外部 OAuth 2.0 ID プロバイダー (IdP) によって管理されます。vhost、交換、キュー、トピックのユーザー認証とリソースアクセス許可は、OAuth 2.0 プロバイダーのスコープシステムを通じて一元化されます。これにより、ユーザー管理が簡素化され、既存の ID システムとの統合が可能になります。詳細については、OAuth 2.0 の認証と認可」を参照してください。

IAM 認証と認可

この方法では、ブローカーユーザーは IAM AWS アウトバウンドフェデレーションを介して IAM 認証情報を使用して認証します。IAM 認証情報は、 AWS Security Token Service (STS) から JWT トークンを取得するために使用されます。これらの JWT トークンは、認証用の OAuth 2.0 トークンとして機能します。このメソッドは、Amazon MQ for RabbitMQ の既存の OAuth 2.0 サポートを活用します。 は OAuth 2.0 ID プロバイダー AWS として機能します。ユーザー認証は IAM AWS によって処理され、vhost、エクスチェンジ、キュー、トピックのリソースアクセス許可は RabbitMQ で設定された IAM ポリシーとスコープエイリアスによって管理されます。詳細については、「IAM 認証と認可」を参照してください。

LDAP 認証と認可

この方法では、ブローカーユーザーとそのアクセス許可は外部 LDAP ディレクトリサービスによって管理されます。ユーザー認証とリソースのアクセス許可は LDAP サーバーを通じて一元化され

るため、ユーザーは既存のディレクトリサービス認証情報を使用して RabbitMQ にアクセスできます。詳細については、「LDAP 認証と認可」を参照してください。

HTTP 認証と認可

この方法では、ブローカーユーザーとそのアクセス許可は外部 HTTP サーバーによって管理されます。ユーザー認証とリソースのアクセス許可は HTTP サーバーを通じて一元化されるため、ユーザーは独自の認証および認可プロバイダーを使用して RabbitMQ にアクセスできます。この方法の詳細については、「HTTP 認証と認可」を参照してください。

SSL 証明書認証

Amazon MQ は、RabbitMQ ブローカーの相互 TLS (mTLS) をサポートしています。SSL 認証プラグインは、mTLS 接続からのクライアント証明書を使用してユーザーを認証します。この方法では、ブローカーユーザーはユーザー名とパスワードの認証情報の代わりに X.509 クライアント証明書を使用して認証されます。クライアントの証明書は信頼できる認証機関 (CA) に対して検証され、ユーザー名は共通名 (CN) やサブジェクト代替名 (SAN) などの証明書のフィールドから抽出されます。この方法は、ネットワーク経由で認証情報を送信せずに強力な認証を提供します。詳細については、「SSL 証明書認証」を参照してください。

> **Note**
>
> RabbitMQ は、同時に使用する複数の認証および認可方法をサポートしています。たとえば、OAuth 2.0 と簡易 (内部) 認証の両方を有効にできます。詳細については、OAuth 2.0 チュートリアルセクション「OAuth 2.0 と簡易 (内部) 認証の両方を有効にする」および「RabbitMQ アクセスコントロールドキュメント」を参照してください。
> Amazon MQ では、認証設定をテストするときに内部ユーザーを作成することをお勧めします。これにより、RabbitMQ 管理 API を使用してアクセス設定を検証できます。詳細については、「アクセス検証」を参照してください。

Amazon MQ for ActiveMQ の認証と認可

Amazon MQ for ActiveMQ は、次の認証および認可方法をサポートしています。

シンプルな認証と認可

この方法では、ブローカーユーザーは Amazon MQ コンソールまたは API を使用して作成および管理されます。ユーザーには、キュー、トピック、および ActiveMQ ウェブコンソールにアクセスする

ための特定のアクセス許可を設定できます。この方法の詳細については、ActiveMQ ブローカーユーザーの作成」を参照してください。

LDAP 認証と認可

この方法では、ブローカーユーザーは LDAP サーバーに保存されている認証情報を使用して認証します。ユーザーを追加、削除、変更し、LDAP サーバーを介してトピックとキューにアクセス許可を割り当てることで、一元的な認証と認可を提供できます。この方法の詳細については、ActiveMQ ブローカーと LDAP の統合」を参照してください。

Amazon MQ ブローカーエンジンバージョンのアップグレード

Amazon MQ は、サポートされているすべてのブローカーエンジンタイプに対して、新しいブローカーエンジンバージョンを定期的に提供します。新しいエンジンバージョンには、セキュリティパッチ、バグ修正、その他のブローカーエンジンの改善が含まれています。

Amazon MQ は、`X.Y.Z` 形式のセマンティックバージョニングに従ってバージョン番号を分類します。Amazon MQ の実装では、`X` はメジャーバージョンを示し、`Y` はマイナーバージョンを表し、`Z` はパッチバージョン番号を示します。Amazon MQ は 2 種類のアップグレードをサポートしています。

- メジャーバージョンアップグレード – メジャーエンジンバージョン番号が変更されたときに行われます。例えば、RabbitMQ バージョン 3.13 からバージョン 4.2 へのアップグレードは、メジャーバージョンのアップグレードと見なされます。
- マイナーバージョンアップグレード – マイナーエンジンバージョン番号のみが変更されたときに行われます。たとえば、バージョン 3.11 からバージョン 3.12 へのアップグレードは、マイナーバージョンアップグレードと見なされます。

ブローカーはいつでも、サポートされている次のメジャーバージョンまたはマイナーバージョンに手動でアップグレードできます。Amazon MQ は、スケジュールされたメンテナンスウィンドウ中に、すべてのブローカーでサポートされている最新のパッチバージョンへのアップグレードを管理します。手動バージョンアップグレードと自動バージョンアップグレードは、スケジュールされたメンテナンスウィンドウ中、またはブローカーを再起動した後に行われます。Amazon MQ は、現在のマイナーバージョンがサポート終了に達すると、ブローカーを次のマイナーバージョンにアップグレードします。

エンジンバージョンの手動アップグレード

ブローカーのエンジンバージョンは、 AWS マネジメントコンソール、、 AWS CLIまたは Amazon MQ API を使用してアップグレードできます。

AWS マネジメントコンソール

を使用してブローカーのエンジンバージョンをアップグレードするには AWS マネジメントコンソール

1. ブローカーの詳細ページで [Edit] (編集) をクリックします。
2. [Specifications] (仕様) の [Broker engine version] (ブローカーエンジンバージョン) で、ドロップダウンリストから新しいバージョン番号を選択します。
3. ページの最下部までスクロールして、[Schedule modifications] (変更をスケジュールする) をクリックします。
4. [Schedule broker modifications] (ブローカー変更のスケジュール) ページの [When to apply modifications] (変更を適用するタイミング) で以下のいずれかを選択します。
   - 次にスケジュールされたメンテナンスウィンドウ中に Amazon MQ でバージョンアップグレードを完了する場合は、[After the next reboot] (次回の再起動後) を選択します。
   - 直ちにブローカーを再起動してエンジンバージョンをアップグレードする場合は、[Immediately] (即時) を選択します。

   > **Important**
   >
   > 再起動中、シングルインスタンスブローカーはオフラインになります。クラスターブローカーの場合、ブローカーの再起動中にダウンするノードは 1 つだけです。
5. [Apply] (適用) をクリックして、変更の適用を終了します。

AWS CLI

を使用してブローカーのエンジンバージョンをアップグレードするには AWS CLI

1. 以下の例にあるように、update-broker CLI コマンドを使用して、以下のパラメータを指定します。

- `--broker-id` – Amazon MQ がブローカー用に生成する一意の ID です。ID は、ブローカー ARN から解析できます。例えば、`arn:aws:mq:us-east-2:123456789012:broker:MyBroker:b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9` という ARN の場合、ブローカー ID は `b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9` になります。
- `--engine-version` – ブローカーエンジンをアップグレードするエンジンバージョン番号です。

```
aws mq update-broker --broker-id broker-id --engine-version version-number
```

2. (オプション) エンジンバージョンをすぐにアップグレードする場合は、reboot-broker CLI コマンドを使用してブローカーを再起動します。

```
aws mq reboot-broker --broker-id broker-id
```

直ちにブローカーを再起動して変更を適用しない場合は、次にスケジュールされたメンテナンスウィンドウ中に Amazon MQ がブローカーをアップグレードします。

> **Important**
>
> 再起動中、シングルインスタンスブローカーはオフラインになります。クラスターブローカーの場合、ブローカーの再起動中にダウンするノードは 1 つだけです。

Amazon MQ API

Amazon MQ API を使用してブローカーのエンジンバージョンをアップグレードする

1. UpdateBroker API オペレーションを使用します。パスパラメータとして `broker-id` を指定します。以下の例は、ブローカーが `us-west-2` リージョンにあることを前提としています。利用可能な Amazon MQ エンドポイントの詳細については、「AWS 全般のリファレンス」の「Amazon MQ エンドポイントとクォータ」を参照してください。

```
PUT /v1/brokers/broker-id HTTP/1.1
Host: mq.us-west-2.amazonaws.com
Date: Mon, 7 June 2021 12:00:00 GMT
x-amz-date: Mon, 7 June 2021 12:00:00 GMT
```

```
Authorization: authorization-string
```

リクエストペイロードで engineVersion を使用して、ブローカーをアップグレードするバージョン番号を指定します。

```
{
    "engineVersion": "engine-version-number"
}
```

2. (オプション) エンジンバージョンをすぐにアップグレードする場合は、RebootBroker API オペレーションを使用してブローカーを再起動します。 broker-idはパスパラメータとして指定されます。

```
POST /v1/brokers/broker-id/reboot-broker HTTP/1.1
Host: mq.us-west-2.amazonaws.com
Date: Mon, 7 June 2021 12:00:00 GMT
x-amz-date: Mon, 7 June 2021 12:00:00 GMT
Authorization: authorization-string
```

直ちにブローカーを再起動して変更を適用しない場合は、次にスケジュールされたメンテナンスウィンドウ中に Amazon MQ がブローカーをアップグレードします。

> **Important**
>
> 再起動中、シングルインスタンスブローカーはオフラインになります。クラスターブローカーの場合、ブローカーの再起動中にダウンするノードは 1 つだけです。

Amazon MQ ブローカーインスタンスタイプのアップグレード

> **Important**
>
> mq.m7g.x インスタンスは、Amazon MQ for RabbitMQ ブローカーでのみ使用できます。Amazon MQ for ActiveMQ ブローカーは mq.m5.x インスタンスのみを使用します。

ブローカーのインスタンスクラス (m7g) およびサイズ (large) を組み合わせた説明は、ブローカーインスタンスタイプ (mq.m7g.large など) と呼ばれます。インスタンスタイプを選択するときは、ブローカーのパフォーマンスに影響する以下の要因を考慮することが重要です。

- クライアントとキューの数
- 送信されるメッセージの量
- メモリに保持されるメッセージ
- 冗長メッセージ

小さいブローカーインスタンスタイプ (mq.m7g.medium) は、アプリケーションのパフォーマンスをテストする場合にのみ使用することをお勧めします。本番稼働レベルのクライアントとキュー、高スループット、メモリ内のメッセージ、冗長メッセージには、大きいブローカーインスタンスタイプ (mq.m7g.large 以上) が推奨されます。

パフォーマンスの問題が発生した場合、またはテストから本番環境に移行する場合は、より大きなインスタンスタイプ (micro から large) にアップグレードすることをお勧めします。インスタンスタイプをアップグレードするには、AWS マネジメントコンソール、AWS CLI、または Amazon MQ API を使用できます。

AWS マネジメントコンソール

AWS マネジメントコンソール を使用してより大きなインスタンスタイプにアップグレードするには、次の手順を実行します。

1. Amazon MQ コンソールにサインインします。
2. 左のナビゲーションペインで、[Brokers] (ブローカー) をクリックしてから、アップグレードするブローカーをリストから選択します。
3. ブローカーの詳細ページで [Edit] (編集) をクリックします。
4. [仕様] の [ブローカーインスタンスタイプ] で、ドロップダウンリストから新しいインスタンスタイプを選択します。
5. ページの最下部で、[変更をスケジュールする] をクリックします。
6. [Schedule broker modifications] (ブローカー変更のスケジュール) ページの [When to apply modifications] (変更を適用するタイミング) で以下のいずれかを選択します。
   - 次にスケジュールされたメンテナンスウィンドウ中に Amazon MQ でアップグレードを完了する場合は、[次回の再起動後] を選択します。

- 直ちにブローカーを再起動してインスタンスタイプをアップグレードする場合は、[即時] を選択します。

> **Important**
>
> 再起動中、シングルインスタンスブローカーはオフラインになります。クラスターブローカーの場合、ブローカーの再起動中にダウンするノードは 1 つだけです。

7. [Apply] (適用) をクリックして、変更の適用を終了します。

AWS CLI

AWS CLI を使用してブローカーのインスタンスタイプをアップグレードする

1. 以下の例にあるように、modify-broker CLI コマンドを使用して、以下のパラメータを指定します。

   - `--broker-id` – Amazon MQ がブローカー用に生成する一意の ID です。
   - `--host-instance-type` – ブローカーエンジンをアップグレードするエンジンバージョン番号です。

```
aws mq modify-broker --broker-id broker-id --host-instance-type instance-type
```

2. (オプション) インスタンスタイプを直ちにアップグレードする場合は、reboot-broker CLI コマンドを使用してブローカーを再起動します。

```
aws mq reboot-broker --broker-id broker-id
```

直ちにブローカーを再起動して変更を適用しない場合は、次にスケジュールされたメンテナンスウィンドウ中に Amazon MQ がブローカーをアップグレードします。

> **Important**
>
> 再起動中、シングルインスタンスブローカーはオフラインになります。クラスターブローカーの場合、ブローカーの再起動中にダウンするノードは 1 つだけです。

Amazon MQ API

Amazon MQ API を使用してブローカーのインスタンスタイプをアップグレードする

1. UpdateBroker API オペレーションを使用します。パスパラメータとして broker-id を指定します。以下の例は、ブローカーが us-west-2 リージョンにあることを前提としています。利用可能な Amazon MQ エンドポイントの詳細については、「AWS 全般のリファレンス」の「Amazon MQ エンドポイントとクォータ」を参照してください。

```
PUT /v1/brokers/broker-id HTTP/1.1
Host: mq.us-west-2.amazonaws.com
Date: Mon, 7 June 2021 12:00:00 GMT
x-amz-date: Mon, 7 June 2021 12:00:00 GMT
Authorization: authorization-string
```

   リクエストペイロードで host-instance-type を使用して、ブローカーをアップグレードするインスタンスタイプを指定します。

```
{
    "host-instance-type": "host-instance-type"
}
```

2. (オプション) エンジンバージョンを直ちにアップグレードする場合は、RebootBroker API オペレーションを使用してブローカーを再起動します。パスパラメータとして broker-id が指定されます。

```
POST /v1/brokers/broker-id/reboot-broker HTTP/1.1
Host: mq.us-west-2.amazonaws.com
Date: Mon, 7 June 2021 12:00:00 GMT
x-amz-date: Mon, 7 June 2021 12:00:00 GMT
Authorization: authorization-string
```

   直ちにブローカーを再起動して変更を適用しない場合は、次にスケジュールされたメンテナンスウィンドウ中に Amazon MQ がブローカーをアップグレードします。

> **Important**
>
> 再起動中、シングルインスタンスブローカーはオフラインになります。クラスターブローカーの場合、ブローカーの再起動中にダウンするノードは 1 つだけです。

Amazon MQ for ActiveMQ ストレージタイプ

Amazon MQ for ActiveMQ は Amazon Elastic File System (EFS) と Amazon Elastic Block Store (EBS) をサポートしています。デフォルトで、ActiveMQ ブローカーはブローカーストレージに Amazon EFS を使用します。複数のアベイラビリティーゾーン全体で優れた耐障害性とレプリケーションを活用するには、Amazon EFS を使用します。低レイテンシーと高スループットを活用するには、Amazon EBS を使用します。

Important

- Amazon EBS を使用できるのは、`mq.m5` ブローカーインスタンスタイプファミリーのみです。
- ブローカーインスタンスタイプを変更することはできますが、ブローカーを作成した後でブローカーストレージタイプを変更することはできません。
- Amazon EBS は単一のアベイラビリティーゾーン内でデータをレプリケートし、ActiveMQ アクティブ/スタンバイデプロイモードをサポートしません。

ストレージタイプ間の相違点

以下の表は、ActiveMQ ブローカー向けのインメモリ、Amazon EFS、および Amazon EBS の各ストレージタイプの違いを簡単にまとめたものです。

ストレージタイプ	永続的	ユースケースの例	プロデューサーあたり、1 秒あたりのエンキューされたメッセージの概算最大数 (1 KB のメッセージ)	レプリケーション
インメモリ	非永続的	• 株価情報 • 位置情報の更新 • 頻繁に変更されるデータ	5,000	なし

ストレージタイプ	永続的	ユースケースの例	プロデューサーあたり、1 秒あたりのエンキューされたメッセージの概算最大数 (1 KB のメッセージ)	レプリケーション
Amazon EBS	永続	• 大量のテキスト • 注文処理	500	1 つのアベイラビリティーゾーン (AZ) 内の複数のコピー
Amazon EFS	永続	金融取引	80	複数の AZ にまたがる複数のコピー

インメモリメッセージストレージは、レイテンシーが最も低く、最大のスループットを提供します。ただし、メッセージはインスタンスの置き換えまたはブローカーの再起動中に失われます。

Amazon EFS は、高い耐久性を備え、複数の AZ にまたがってレプリケートされるように設計されており、単一のコンポーネントの障害、または AZ の可用性に影響する問題が原因で発生するデータの損失を防ぎます。Amazon EBS はスループット用に最適化されており、単一の AZ 内にある複数のサーバー全体にレプリケートされます。

プライベート Amazon MQ ブローカーの設定

プライベートブローカーにはパブリックアクセシビリティがないため、VPC の外部からアクセスすることはできません。プライベートブローカーを設定する前に、VPC、サブネット、セキュリティグループに関する次の情報を確認してください。

- VPC
  - ブローカーのサブネットとセキュリティグループは、同じ VPC 内にある必要があります。
  - プライベートブローカーを使用している場合は、VPC で設定していない IP アドレスが表示されることがあります。これらは Amazon MQ インフラストラクチャ上の IP アドレスであり、対応は必要ありません。

- サブネット
  - サブネットが共有 VPC 内にある場合、VPC はブローカーを作成する同じアカウントによって所有されている必要があります。
  - サブネットが指定されていない場合は、デフォルト VPC のデフォルトサブネットが使用されます。
  - ブローカーが作成されると、使用されたサブネットを変更することはできません。
  - クラスターおよびアクティブ/スタンバイブローカーの場合、サブネットは異なるアベイラビリティーゾーンにある必要があります。
  - 単一インスタンスブローカーの場合、使用するサブネットを指定し、ブローカーを同じアベイラビリティーゾーン内に作成できます。
- セキュリティグループ
  - セキュリティグループが指定されていない場合は、デフォルトの VPC のデフォルトのセキュリティグループが使用されます。
  - 単一インスタンス、クラスター、アクティブ/スタンバイブローカーは、少なくとも 1 つのセキュリティグループが必要です (例: デフォルトのセキュリティグループ)。

    > **Note**
    >
    > パブリック RabbitMQ ブローカーは、サブネットやセキュリティグループを使用しません。

  - ブローカーが作成されると、使用されたセキュリティグループは変更できません。セキュリティグループ自体は引き続き変更できます。

でのプライベートブローカーの設定 AWS マネジメントコンソール

プライベートブローカーを設定するには、 AWS マネジメントコンソールで新しいブローカーの作成を開始します。次に、[ネットワーク設定] セクションで、ブローカーの接続を設定するには、以下を実行します。

1. ブローカーの [プライベートアクセス] を選択します。プライベートブローカーに接続するには、IPv4, IPv6、またはデュアルスタック (IPv4 および IPv6) を使用できます。詳細については、「Connecting to Amazon MQ」を参照してください。
2. 次に、[デフォルトの VPC、サブネット (複数可)、セキュリティグループ (複数可) を使用する] を選択するか、[既存の VPC、サブネット (複数可)、セキュリティグループ (複数可) の選択] を

選択します。デフォルトまたは既存の VPC、サブネット (複数可)、またはセキュリティグループ (複数可) を使用しない場合は、プライベートブローカーに接続する新しい VPC を作成する必要があります。

> **Note**
>
> プライベートブローカーアクセスの場合、接続方法はサブネットの選択した IP タイプと同じになります。ブローカーが作成されると、VPC エンドポイントは変更できず、常に選択したサブネットの IP タイプになります。新しい IP タイプを使用する場合は、新しいブローカーを作成する必要があります。

> **Note**
>
> Amazon MQ for ActiveMQ は VPC エンドポイントを使用しません。ActiveMQ ブローカーを初めて作成すると、Amazon MQ は VPC で Elastic Network Interface (ENI) をプロビジョニングします。セキュリティグループは ENI に配置され、パブリックブローカーとプライベートブローカーの両方に使用できます。

パブリックアクセシビリティのない Amazon MQ ブローカーのウェブコンソールへのアクセス

ブローカーのパブリックアクセシビリティをオフにすると、ブローカーを作成した AWS アカウント ID がプライベートブローカーにアクセスできます。ブローカーのパブリックアクセシビリティを無効にしている場合、ブローカーのウェブコンソールにアクセスするには、以下の手順を実行する必要があります。

1. `public-vpc` に Linux EC2 インスタンスを作成します (必要に応じて、パブリック IP を使用)。
2. VPC が正しく設定されていることを確認するには、作成した EC2 インスタンスへの `ssh` 接続を確立し、ブローカーの URI を指定して `curl` コマンドを使用します。
3. お使いのマシンから、プライベートキーファイルのパスとパブリック EC2 インスタンスの IP アドレスを使用して、EC2 インスタンスへの `ssh` トンネルを作成します。以下はその例です。

```
ssh -i ~/.ssh/id_rsa -N -C -q -f -D 8080 ec2-user@203.0.113.0
```

転送プロキシサーバーがマシン上で開始されます。

4. プロキシクライアント (例: FoxyProxy) をマシン上にインストールします。
5. 以下の設定を使用して、プロキシクライアントを設定します。

   - プロキシタイプで、SOCKS5 を指定します。
   - IP アドレス、DNS 名、サーバー名で、localhost を指定します。
   - ポートで、8080 を指定します。
   - 既存の URL パターンをすべて削除します。
   - URL パターンで、*.mq.*.amazonaws.com* を指定します。
   - 接続タイプで、HTTP(S) を指定します。

   プロキシクライアントを有効にすると、マシン上のウェブコンソールにアクセスできます。

> **Important**
>
> プライベートブローカーを使用している場合は、VPC で設定していない IP アドレスが表示されることがあります。これらは Amazon MQ インフラストラクチャ上の RabbitMQ からの IP アドレスであり、対応は必要ありません。

Amazon MQ ブローカーのメンテナンスウィンドウのスケジュール

Amazon MQ は、ハードウェア、オペレーティングシステム、またはメッセージブローカーのエンジンソフトウェアに対して、メンテナンスウィンドウ内で定期的にメンテナンスを実行します。例えば、ブローカーインスタンスタイプを変更した場合、Amazon MQ はスケジュールされた次回のメンテナンスウィンドウ内で変更を適用します。メンテナンスの所要時間は、メッセージブローカーにスケジュールされている操作に応じて最大 2 時間かかることがあります。複数のアベイラビリティーゾーン (AZ) にまたがる高可用性のブローカーデプロイモードを選択すると、メンテナンスウィンドウ中のダウンタイムを最小限に抑えることができます。

Amazon MQ for ActiveMQ には、高可用性を実現するアクティブ/スタンバイデプロイが用意されています。アクティブ/スタンバイモードでは、Amazon MQ はメンテナンス操作を一度に 1 インスタンスずつ実行するため、少なくとも 1 つのインスタンスは利用可能な状態に維持されます。さらに、メンテナンスウィンドウが 1 週間の異なる時点に設定されたブローカーのネットワークを構成することもできます。Amazon MQ for RabbitMQ には、高可用性を実現するクラスターデプロイが

用意されています。クラスターデプロイでは、Amazon MQ はメンテナンス操作を一度に 1 ノードずつ実行して、少なくとも 2 つのノードが常に稼働している状態を維持します。

ブローカーを最初に作成するときは、メンテナンスウィンドウを週に 1 回、指定時刻に実行するようにスケジュールできます。ブローカーのメンテナンスウィンドウは、次にスケジュールされたメンテナンスウィンドウまで、最大 4 回しか調整できません。ブローカーのメンテナンスウィンドウが完了すると、Amazon MQ はこの制限をリセットし、次回のメンテナンスウィンドウの実行前に再びスケジュールを調整できるようになります。ブローカーの可用性は、ブローカーのメンテナンスウィンドウを調整しても影響を受けません。

ブローカーメンテナンスウィンドウを調整するには、AWS マネジメントコンソール、AWS CLI、または Amazon MQ API を使用できます。

AWS マネジメントコンソールを使用したブローカーメンテナンスウィンドウのスケジュール

AWS マネジメントコンソールを使用してブローカーのメンテナンスウィンドウを調整する

1. Amazon MQ コンソールにサインインします。
2. 左のナビゲーションペインで、[Brokers] (ブローカー) をクリックしてから、アップグレードするブローカーをリストから選択します。
3. ブローカーの詳細ページで [Edit] (編集) をクリックします。
4. [Maintenance] (メンテナンス) で、以下を実行します。

   a. [Start day (開始日)] には、ドロップダウンリストから曜日を選択します ([Sunday (日曜日)] など)。

   b. [Start time (開始時刻)] には、次回のブローカーメンテナンスウィンドウをスケジュールする時間と分を選択します (12:00 など)。

   > **Note**
   >
   > [Start time] (開始時刻) オプションは、UTC+0 タイムゾーンで設定されます。

5. 次に、[変更のスケジュール] を選択します。[次回の再起動後] または [即時] を選択します。[次回の再起動後] を選択すると、ブローカーを再起動せずに、直ちにメンテナンスウィンドウが更新されます。[即時] を選択すると、ブローカーがすぐに再起動されます。
6. ブローカーの詳細ページにある [Maintenance window (メンテナンスウィンドウ)] で、希望する新しいスケジュールが表示されていることを確認します。

AWS CLI を使用したブローカーメンテナンスウィンドウのスケジュール

AWS CLI を使用してブローカーメンテナンスウィンドウを調整する

1. 以下の例にあるように、update-broker CLI コマンドを使用して、以下のパラメータを指定します。

   - `--broker-id` – Amazon MQ がブローカー用に生成する一意の ID です。ID は、ブローカー ARN から解析できます。例えば、`arn:aws:mq:us-east-2:123456789012:broker:MyBroker:b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9` という ARN の場合、ブローカー ID は `b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9` になります。
   - `--maintenance-window-start-time` – 以下の構造で提供される、週次メンテナンスウィンドウの開始時刻を決定するパラメータです。
     - `DayOfWeek` – の構文で指定する曜日です。`MONDAY| TUESDAY | WEDNESDAY | THURSDAY | FRIDAY | SATURDAY | SUNDAY`
     - `TimeOfDay` – 24 時間形式の時刻です。
     - `TimeZone` – (オプション) 国/都市、または UTC オフセット形式のいずれかで指定するタイムゾーンです。デフォルトで UTC に設定されます。

```
aws mq update-broker --broker-id broker-id \
--maintenance-window-start-time DayOfWeek=SUNDAY,TimeOfDay=13:00,TimeZone=America/Los_Angeles
```

2. (オプション) describe-broker CLI コマンドを使用して、メンテナンスウィンドウが正常に更新されたことを検証します。

```
aws mq describe-broker --broker-id broker-id
```

Amazon MQ API を使用したブローカーメンテナンスウィンドウのスケジュール

Amazon MQ API を使用してブローカーメンテナンスウィンドウを調整する

1. UpdateBroker API オペレーションを使用します。パスパラメータとして `broker-id` を指定します。以下の例は、ブローカーが `us-west-2` リージョンにあることを前提としています。

利用可能な Amazon MQ エンドポイントの詳細については、「AWS 全般のリファレンス」の「Amazon MQ エンドポイントとクォータ」を参照してください。

```
PUT /v1/brokers/broker-id HTTP/1.1
Host: mq.us-west-2.amazonaws.com
Date: Wed, 7 July 2021 12:00:00 GMT
x-amz-date: Wed, 7 July 2021 12:00:00 GMT
Authorization: authorization-string
```

リクエストペイロードには、maintenanceWindowStartTime パラメータと WeeklyStartTime リソースタイプを使用します。

```
{
"maintenanceWindowStartTime": {
    "dayOfWeek": "SUNDAY",
    "timeZone": "America/Los_Angeles",
    "timeOfDay": "13:00"
  }
}
```

2. (オプション) DescribeBroker API オペレーションを使用して、メンテナンスウィンドウが正常に更新されたことを検証します。パスパラメータとして broker-id が指定されています。

```
GET /v1/brokers/broker-id HTTP/1.1
Host: mq.us-west-2.amazonaws.com
Date: Wed, 7 July 2021 12:00:00 GMT
x-amz-date: Wed, 7 July 2021 12:00:00 GMT
Authorization: authorization-string
```

Amazon MQ ブローカーの再起動

新しい設定をブローカーに適用するには、ブローカーを再起動します。

> **Note**
>
> ActiveMQ ブローカーが応答しない場合、ブローカーを再起動して障害状態から復旧できます。

以下の例では、 AWS マネジメントコンソールを使用して Amazon MQ ブローカーを再起動する方法を説明します。

Amazon MQ ブローカーを再起動する

1. Amazon MQ コンソールにサインインします。
2. ブローカーのリストから、ブローカーの名前 (MyBroker など) を選択します。
3. [***MyBroker***] ページで、[Actions]、[Reboot broker] の順に選択します。

   > **Important**
   >
   > 再起動中、シングルインスタンスブローカーはオフラインになります。クラスターブローカーは利用できますが、各ノードは一度に 1 つずつ再起動されます。

4. [Reboot broker] ダイアログボックスで、[Reboot] を選択します。

   Rebooting a broker takes about 5 minutes.」が表示されます。再起動にインスタンスサイズの変更が含まれているか、キューの深さが大きいブローカーで再起動が実行される場合、再起動プロセスに時間がかかることがあります。

Amazon MQ ブローカーの削除

Amazon MQ ブローカーを使用しない (および近い将来使用しない) 場合は、Amazon MQ から削除して AWS コストを削減するのがベストプラクティスです。

以下の例では、 AWS マネジメントコンソールを使用してブローカーを削除する方法を説明します。

Amazon MQ ブローカーの削除

1. Amazon MQ コンソールにサインインします。
2. ブローカーのリストからブローカー (MyBroker など) を選択して、[Delete] (削除) をクリックします。
3. [Delete ***MyBroker***?] (MyBroker を削除しますか?) ダイアログボックスで、`delete` と入力してから [Delete] (削除) をクリックします。

   ブローカーの削除には約 5 分かかります。

Amazon MQ ブローカーのステータス

ステータスによって、ブローカーの現在の状態が示されます。以下の表には、Amazon MQ ブローカーのステータスがリストされています。

コンソール	API	説明
作成に失敗	CREATION_FAILED	ブローカーを作成できませんでした。
作成を実行中	CREATION_IN_PROGRESS	ブローカーは現在作成中です。
削除を実行中	DELETION_IN_PROGRESS	ブローカーは現在削除中です。
再起動の進行中	REBOOT_IN_PROGRESS	ブローカーは現在再起動中です。
実行中	RUNNING	ブローカーが機能しています。
重要なアクションは不要	CRITICAL_ACTION_REQUIRED	ブローカーは実行中ですが、パフォーマンスが低下した状態にあり、即時の処置が必要です。問題を解決する手順については、トラブルシューティング のリストからアクション必須コードを選択します。

Amazon MQ リソースへのタグの追加

コスト割り当てのために Amazon MQ リソースを分類して識別するには、ブローカーまたは設定の目的を特定するメタデータタグを追加できます。これはブローカーが多数ある場合に特に便利です。コスト配分タグを使用してAWSの請求書を整理し、自分のコスト構造に反映できます。そのため

には、サインアップして AWS アカウントの請求書にタグキーおよび値を含めます。詳細については、AWS Billing ユーザーガイドの「月別コスト配分レポートの設定」を参照してください。

例えば、コストセンターと、Amazon MQ リソースの目的を表すタグを追加できます。

リソース	キー	値
Broker1	Cost Center	34567
	Stack	Production
Broker2	Cost Center	34567
	Stack	Production
Broker3	Cost Center	12345
	Stack	Development

このタグ付けスキームでは、同じコストセンター内で関連するタスクを実行している 2 つのブローカーをグループ化しながら、関連しないブローカーに異なるコスト割り当てタグを付けることができます。

Amazon MQ コンソールでのタグの追加

次の手順に従うと、Amazon MQ コンソールでリソースを作成しているときにすばやくタグを追加できます。

1. [ブローカーの作成] ページで、[追加設定] を選択します。
2. [タグ] で、[タグの追加] を選択します。
3. [キー] と [値] のペアを入力します。
4. (オプション) [タグの追加] を選択して、ブローカーに複数のタグを追加します。
5. [バケットを作成する] を選択します。

設定を作成するときにタグを追加するには。

1. [設定の作成] ページで、[アドバンスト] を選択します。
2. [タグ] を選択し、[設定の作成] ページで、[タグの追加] を選択します。

3. [キー] と [値] のペアを入力します。
4. (オプション) [タグの追加] を選択して、設定に複数のタグを追加します。
5. [起動設定の作成] を選択します。

タグを追加した後、Amazon MQ コンソールでリソースのタグを表示、編集、削除できます。REST API を使用してリソースのタグを確認することもできます。詳細については、Amazon MQ REST API リファレンスを参照してください。

Amazon MQ for ActiveMQ の使用

Amazon MQ は、ニーズに適したコンピューティングおよびストレージリソースを使用したメッセージブローカーの作成を容易にします。ブローカーは、AWS マネジメントコンソール、Amazon MQ REST API、または AWS Command Line Interface を使用して作成、管理、および削除することができます。

Amazon MQ for ActiveMQ ブローカーは、単一インスタンスブローカーまたはアクティブ/スタンバイブローカーとしてデプロイできます。どちらのデプロイモードでも、Amazon MQ はデータを冗長的に保存することによって優れた耐久性を提供します。

> **Note**
>
> Amazon MQ は、データストアとして Apache KahaDB を使用します。JDBC および LevelDB などの他のデータストアはサポートされていません。

ブローカーには、ActiveMQ がサポートする任意のプログラミング言語を使用し、以下のプロトコルに対して TLS を明示的に有効にすることによってアクセスできます。

- AMQP
- MQTT
- MQTT over WebSocket
- OpenWire
- STOMP
- STOMP over WebSocket

Amazon MQ REST API については、Amazon MQ REST API リファレンスを参照してください。

Amazon MQ for ActiveMQ ブローカー

Amazon MQ for ActiveMQ ブローカーとは

ブローカーは、Amazon MQ で実行されるメッセージブローカー環境です。これは、Amazon MQ の基本的な構成要素です。ブローカーインスタンスのクラス (`m5`) およびサイズ (`large`、`medium`) を

組み合わせた説明は、ブローカーインスタンスタイプ (`mq.m5.large` など) と呼ばれます。詳細については、「Broker instance types」を参照してください。

- 単一インスタンスブローカーは、1 つのアベイラビリティーゾーン内の 1 つのブローカーで構成されます。ブローカーは、アプリケーション、および Amazon EBS または Amazon EFS ストレージボリュームと通信します。
- アクティブ/スタンバイブローカーは、2 つの異なるアベイラビリティーゾーンにある 2 つのブローカーで構成され、冗長ペアで設定されます。これらのブローカーは、アプリケーションおよび Amazon EFS と同期的に通信します。

詳細については、「Amazon MQ for ActiveMQ ブローカーのデプロイオプション」を参照してください。

自動マイナーバージョンアップグレードを有効にして、Apache から新しいバージョンがリリースされるたびに、ブローカーエンジンの新しいマイナーバージョンにアップグレードできます。自動アップグレードは、曜日、時刻 (24 時間形式)、およびタイムゾーン (デフォルトは UTC) で定義されたメンテナンスウィンドウ中に行われます。

ブローカーを作成および管理する方法については、以下を参照してください。

- 開始方法: ActiveMQ ブローカーの作成と接続
- ブローカー
- Broker statuses

サポートされているワイヤレベルプロトコル

ブローカーには、ActiveMQ がサポートする任意のプログラミング言語を使用し、以下のプロトコルに対して TLS を明示的に有効にすることによってアクセスできます。

- AMQP
- MQTT
- MQTT over WebSocket
- OpenWire
- STOMP
- STOMP over WebSocket

属性

ActiveMQ ブローカー設定にはいくつかの属性があります。以下はその例です。

- 名前 (`MyBroker`)
- ID (`b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9`)
- Amazon リソースネーム (ARN) (`arn:aws:mq:us-east-2:123456789012:broker:MyBroker:b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9`)
- ActiveMQ ウェブコンソール URL (`https://b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-east-2.amazonaws.com:8162`)

  詳細については、Apache ActiveMQ ドキュメントの「Web Console」を参照してください。

  > **Important**
  >
  > `activemq-webconsole` グループが含まれない認可マップを指定する場合、Amazon MQ ブローカーにメッセージを送信する権限、またはブローカーからメッセージを受信する権限がグループにないことから、ActiveMQ ウェブコンソールは使用できません。

- ワイヤレベルプロトコルのエンドポイント:
  - `amqp+ssl://b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-east-2.amazonaws.com:5671`
  - `mqtt+ssl://b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-east-2.amazonaws.com:8883`
  - `ssl://b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-east-2.amazonaws.com:61617`

    > **Note**
    >
    > これは OpenWire エンドポイントです。

  - `stomp+ssl://b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-east-2.amazonaws.com:61614`
  - `wss://b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-east-2.amazonaws.com:61619`

詳細については、Apache ActiveMQ ドキュメントの「Configuring Transports」を参照してください。

> **Note**
>
> アクティブ/スタンバイブローカーの場合、Amazon MQ は 2 つの ActiveMQ ウェブコンソール URL を提供しますが、一度に 1 つの URL しかアクティブになりません。同様に、Amazon MQ はワイヤレベルプロトコルごとに 2 つのエンドポイントを提供しますが、ペアごとに一度に 1 つのエンドポイントしかアクティブになりません。`-1` および `-2` サフィックスは冗長ペアを表します。

ブローカー属性の完全なリストについては、Amazon MQ REST API リファレンスで以下を参照してください。

- REST 操作 ID: ブローカー
- REST 操作 ID: ブローカー
- REST 操作 ID: ブローカーの再起動

ブローカーユーザー

ActiveMQ ユーザーとは、ActiveMQ ブローカーのキューとトピックにアクセスできる人物またはアプリケーションです。ユーザーは、特定の許可を持つように設定できます。例えば、一部のユーザーに ActiveMQ ウェブコンソールへのアクセスを許可することができます。

グループはセマンティックラベルです。グループをユーザーに割り当てて、グループが特定のキューとトピックに対する送信、受信、管理を行うための許可を設定できます。

> **Important**
>
> ユーザーを変更しても、その変更はユーザーに直ちに適用されません。変更を適用するには、次のメンテナンスウィンドウまで待機するか、ブローカーを再起動する必要があります。

ユーザーとグループの詳細については、Apache ActiveMQ ドキュメントの以下の項目を参照してください。

- Authorization
- 認可の例

ActiveMQ ユーザーを作成、編集および削除する方法については、以下を参照してください。

- ActiveMQ ブローカーユーザーの作成
- Users

ユーザー属性

ユーザー属性の完全なリストについては、Amazon MQ REST API リファレンスで以下を参照してください。

- REST オペレーション ID: ユーザー
- REST オペレーション ID: ユーザー

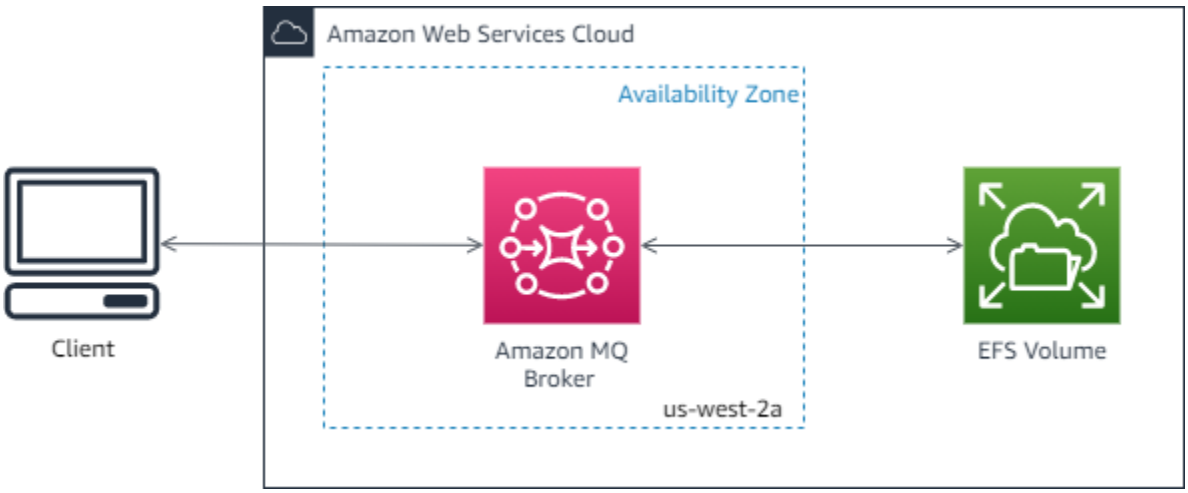
Amazon MQ for ActiveMQ ブローカーのデプロイオプション

Amazon MQ では、ブローカーのデプロイオプションとして、単一インスタンスのデプロイとクラスターデプロイが用意されています。

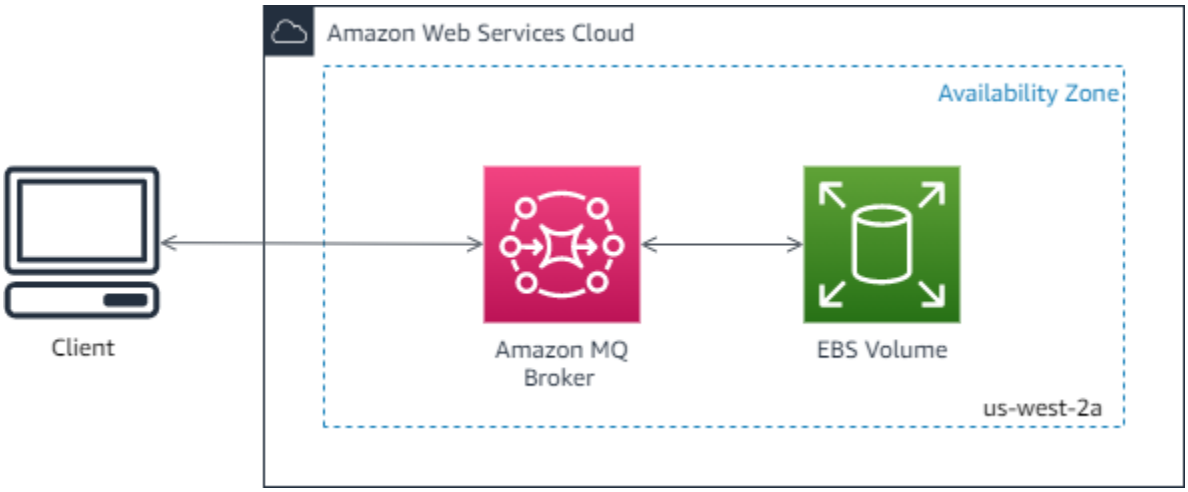
オプション 1: Amazon MQ 単一インスタンスブローカー

単一インスタンスブローカーは、1 つのアベイラビリティーゾーン内の 1 つのブローカーで構成されます。ブローカーは、アプリケーション、および Amazon EBS または Amazon EFS ストレージボリュームと通信します。Amazon EFS ストレージボリュームは、複数のアベイラビリティーゾーン (AZ) にまたがってデータを冗長的に保存することにより、最高レベルの耐久性と可用性を実現するように設計されています。Amazon EBS は、低レイテンシーと高スループット向けに最適化されたブロックレベルのストレージを提供します。ストレージオプションの詳細については、「Storage」を参照してください。

以下の図は、複数の AZ にまたがってレプリケートされている Amazon EFS ストレージを備えた単一インスタンスブローカーを図示しています。



以下の図は、単一の AZ 内にある複数のサーバーにまたがってレプリケートされている Amazon EBS ストレージを備えた単一インスタンスブローカーを図示しています。



オプション 2: 高可用性対応の Amazon MQ アクティブ/スタンバイブローカー

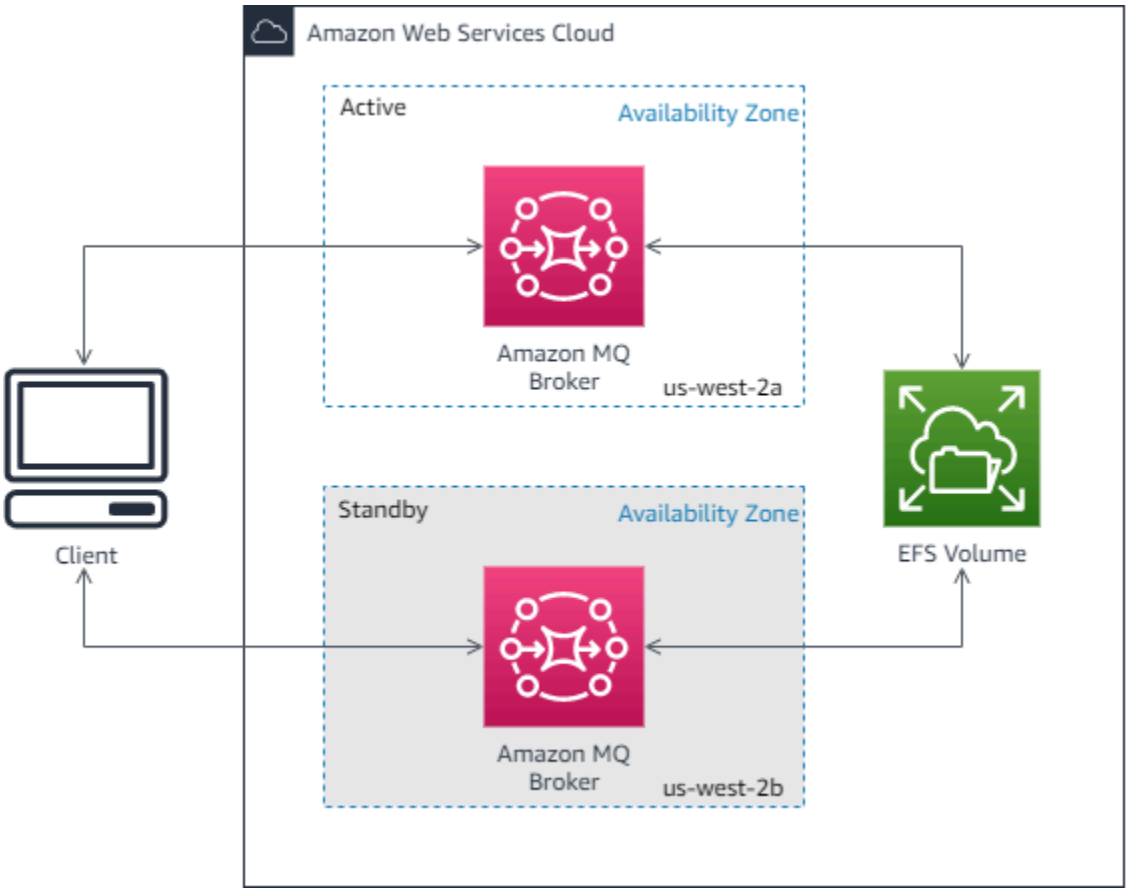
アクティブ/スタンバイブローカーは、2 つの異なるアベイラビリティーゾーンにある 2 つのブローカーで構成され、冗長ペアで設定されます。これらのブローカーは、アプリケーションおよび Amazon EFS と同期的に通信します。Amazon EFS ストレージボリュームは、複数のアベイラビリティーゾーン (AZ) にまたがってデータを冗長的に保存することにより、最高レベルの耐久性と可用性を実現するように設計されています。詳細については、「Storage」を参照してください。

通常、1 つのブローカーインスタンスのみが常時アクティブであり、他のブローカーインスタンスはスタンバイです。ブローカーインスタンスのいずれかが正常に機能しない、またはメンテナンスが行われる場合、Amazon MQ が非アクティブインスタンスを使用停止状態にするまでしばらく時間がか

かります。その間に、正常なスタンバイインスタンスがアクティブになり、着信通信の受け入れを開始できるようになります。メンテナンスウィンドウとブローカーの再起動を開始すると、フェイルオーバーが発生します。ブローカーを再起動する場合、フェイルオーバーには数秒しかかかりません。

アクティブ/スタンバイブローカーの場合、Amazon MQ は 2 つの ActiveMQ ウェブコンソール URL を提供しますが、一度に 1 つの URL しかアクティブになりません。同様に、Amazon MQ はワイヤレベルプロトコルごとに 2 つのエンドポイントを提供しますが、ペアごとに一度に 1 つのエンドポイントしかアクティブになりません。-1 および -2 サフィックスは冗長ペアを表します。ワイヤレベルプロトコルのエンドポイントについては、フェイルオーバートランスポートを使用することによって、アプリケーションがエンドポイントのどちらか一方に接続することを許可する必要があります。

以下の図は、複数の AZ にまたがってレプリケートされている Amazon EFS ストレージを備えたアクティブ/スタンバイブローカーを図示しています。



Amazon MQ のブローカーのネットワーク

Amazon MQ は ActiveMQ のブローカーのネットワーク機能をサポートしています。

ブローカーのネットワークは、同時にアクティブな複数の単一インスタンスブローカー、またはアクティブ/スタンバイブローカーで構成されています。ブローカーのネットワークを作成すると、複数のブローカーインスタンスで可用性、耐障害性、ロードバランシングを向上させることができます。

ブローカーのネットワークの仕組み

ブローカーのネットワークは、ネットワークコネクターを使用してあるブローカーを別のブローカーに接続することによって確立されます。ネットワークコネクターは、あるブローカーから別のブローカーへのオンデマンドメッセージを提供します。ネットワークコネクターは、ブローカー設定で非二重接続または二重接続として設定されます。非二重接続の場合、メッセージは一方のブローカーから他方のブローカーにのみ転送されます。二重接続の場合、メッセージは両方のブローカー間で双方向に転送されます。

ネットワークコネクターが二重として設定されている場合も、メッセージは Broker2 から Broker1 に転送されます。

ブローカーのネットワークでは、非二重接続と二重接続の両方を使用できます。トラフィックを改善したり、制限の引き上げを回避したりするために、別のブローカーに二重接続を導入することもできます。二重接続は、オンプレミスから Amazon MQ マネージドブローカーへの部分的な移行にも役立ちます。

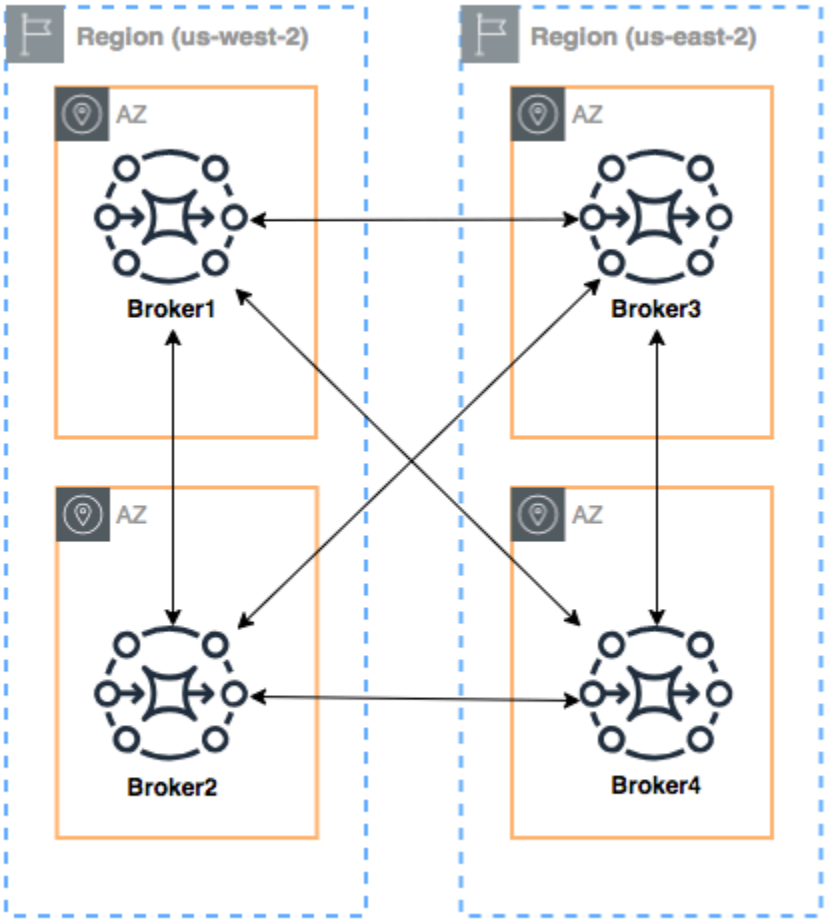
ブローカーのネットワークはどのように認証情報を処理しますか?

ブローカー A がネットワーク内でブローカー B に接続するには、ブローカー A が他のプロデューサーまたはコンシューマーと同様に有効な認証情報を使用する必要があります。ブローカー A の `<networkConnector>` 設定でパスワードを提供する代わりに、ブローカー B の別のユーザーと同じ値を持つブローカー A のユーザーを最初に作成する必要があります (これらは同じユーザー名を共有する別の、一意のユーザーです)。`<networkConnector>` 設定で `userName` 属性を指定すると、Amazon MQ は実行時にパスワードを自動的に追加します。

> **Important**
>
> `<networkConnector>` には `password` 属性を指定しないでください。パスワードが Amazon MQ コンソールに表示されてしまうため、プレーンテキストのパスワードをブローカー設定ファイルに保存することは推奨されません。詳細については、「Configure Network Connectors for Your Broker」を参照してください。

クロスリージョン

AWS リージョンにまたがるブローカーのネットワークを設定するには、それらのリージョンにブローカーをデプロイし、それらのブローカーのエンドポイントへのネットワークコネクタを設定します。



この例のようなブローカーのネットワークを設定するには、これらのブローカーのワイヤレベルのエンドポイントを参照する Broker1 と Broker4 の設定に `networkConnectors` エントリを追加できます。

Broker1 のネットワークコネクター:

```
<networkConnectors>
    <networkConnector name="1_to_2" userName="myCommonUser" duplex="true"
        uri="static:(ssl://b-9876l5k4-32ji-109h-8gfe-7d65c4b132a1-2.mq.us-
west-2.amazonaws.com:61617)"/>
```

```
    <networkConnector name="1_to_3" userName="myCommonUser" duplex="true"
        uri="static:(ssl://b-743c885d-2244-4c95-af67-a85017ff234e-3.mq.us-
east-2.amazonaws.com:61617)"/>
    <networkConnector name="1_to_4" userName="myCommonUser" duplex="true"
        uri="static:(ssl://b-62a7fb31-d51c-466a-a873-905cd660b553-4.mq.us-
east-2.amazonaws.com:61617)"/>
</networkConnectors>
```

Broker2 のネットワークコネクター:

```
<networkConnectors>
    <networkConnector name="2_to_3" userName="myCommonUser" duplex="true"
        uri="static:(ssl://b-743c885d-2244-4c95-af67-a85017ff234e-3.mq.us-
east-2.amazonaws.com:61617)"/>
</networkConnectors>
```

Broker4 のネットワークコネクター:

```
<networkConnectors>
    <networkConnector name="4_to_3" userName="myCommonUser" duplex="true"
        uri="static:(ssl://b-743c885d-2244-4c95-af67-a85017ff234e-3.mq.us-
east-2.amazonaws.com:61617)"/>
    <networkConnector name="4_to_2" userName="myCommonUser" duplex="true"
        uri="static:(ssl://b-9876l5k4-32ji-109h-8gfe-7d65c4b132a1-2.mq.us-
west-2.amazonaws.com:61617)"/>
</networkConnectors>
```

トランスポートコネクタを使用した動的なフェイルオーバー

networkConnector 要素の設定に加えて、ブローカーの transportConnector オプションを設定して動的なフェイルオーバーを有効にし、ネットワークからブローカーが追加または削除されたときに接続を再分散することができます。

```
<transportConnectors>
  <transportConnector name="openwire" updateClusterClients="true"
 rebalanceClusterClients="true" updateClusterClientsOnRemove="true"/>
</transportConnectors>
```

この例では、updateClusterClients および rebalanceClusterClients の両方が true に設定されます。この場合、クライアントにはネットワークのブローカーのリストが提供され、新しいブローカーが参加した場合は再分散がリクエストされます。

利用可能なオプション:

- updateClusterClients: ブローカートポロジのネットワークの変化に関する情報をクライアントに渡します。
- rebalanceClusterClients: 新しいブローカーがブローカーのネットワークに追加されたときに、クライアントはブローカー間で再分散されます。
- updateClusterClientsOnRemove: ブローカーがブローカーのネットワークを離れるときに、トポロジ情報を使用してクライアントを更新します。

updateClusterClients を true に設定すると、クライアントはブローカーのネットワークの 1 つのブローカーに接続するように設定されます。

```
failover:(ssl://b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-
east-2.amazonaws.com:61617)
```

新しいブローカーが接続すると、そのブローカーはネットワーク内のすべてのブローカーの URI のリストを受け取ります。ブローカーへの接続に失敗した場合、接続時に、提供されたいずれかのブローカーに動的に切り替えることができます。

フェイルオーバーの詳細については、Active MQ ドキュメントの「Broker-side Options for Failover」を参照してください。

Amazon MQ for ActiveMQ ブローカーインスタンスタイプ

ブローカーインスタンスのクラス (m5) およびサイズ (large、medium) を組み合わせた説明は、ブローカーインスタンスタイプ (mq.m5.large など) と呼ばれます。次の表は、ActiveMQ ブローカーの使用できる Amazon MQ ブローカーインスタンスタイプの一覧です。

Amazon MQ は、インスタンスタイプがサポート終了に達する少なくとも 90 日前に通知を送信します。中断を防ぐために、サポート終了日より前にブローカーを新しいインスタンスタイプにアップグレードすることをお勧めします。

> **Important**
>
> 2025 年 3 月 17 日以降は、t2.micro または mq.m4.large でブローカーを作成できません。

インスタンスタイプ	vCPU	メモリ (GiB)	推奨用途	収納家具	Amazon MQ でのサポート終了
mq.t3.micro	2	1	評価	EFS	
mq.m5.large	2	8	本番稼働用	EFS または EBS	
mq.m5.xlarge	4	16	本番稼働用	EFS または EBS	
mq.m5.2xlarge	8	32	本番稼働用	EFS または EBS	
mq.m5.4xlarge	16	64	本番稼働用	EFS または EBS	

スループットの考察に関する詳細は、「最良なスループットのために正しいブローカーインスタンスタイプを選択する」を参照してください。

Amazon MQ for ActiveMQ ブローカーの設定

設定には、ActiveMQ ブローカーのすべての設定が XML 形式で含まれています (ActiveMQ の `activemq.xml` ファイルに似ています)。設定は、ブローカーを作成する前に作成することができます。作成後、設定を 1 つ、または複数のブローカーに適用できます。

> **Important**
>
> 設定を変更しても、その変更はブローカーに直ちに適用されません。変更を適用するには、次のメンテナンスウィンドウまで待機するか、ブローカーを再起動する必要があります。設定は `DeleteConfiguration` API を使用してのみ削除できます。詳細については、「Amazon MQ API リファレンス」の「設定」を参照してください。

属性

ブローカー設定には複数の属性があります。次に例を示します。

- 名前 (MyConfiguration)
- ID (c-1234a5b6-78cd-901e-2fgh-3i45j6k178l9)
- Amazon リソースネーム (ARN) (arn:aws:mq:us-east-2:123456789012:configuration:c-1234a5b6-78cd-901e-2fgh-3i45j6k178l9)

設定属性の完全なリストについては、Amazon MQ REST API リファレンスで以下を参照してください。

- REST オペレーション ID: 設定
- REST オペレーション ID: 設定

設定のリビジョン属性の詳細なリストについては、以下を参照してください。

- REST オペレーション ID: 設定のリビジョン
- REST オペレーション ID: 設定のリビジョン

Spring XML 設定ファイルの使用

Spring XML ファイルを使用して ActiveMQ ブローカーを設定します。事前定義された送信先、送信先ポリシー、認可ポリシー、およびプラグインなど、ActiveMQ ブローカーのさまざまな側面を設定できます。Amazon MQ は、ネットワーク転送およびストレージなど、これらの設定要素の一部を制御します。ブローカーのネットワーク作成など、他の設定オプションは、現在サポートされていません。

サポートされている設定オプションの完全なセットは、Amazon MQ XML スキーマに指定されています。次のリンクを使用して、サポートされているスキーマの zip ファイルをダウンロードします。

- amazon-mq-active-mq-5.19.1.xsd.zip
- amazon-mq-active-mq-5.18.4.xsd.zip
- amazon-mq-active-mq-5.17.6.xsd.zip
- amazon-mq-active-mq-5.16.7.xsd.zip
- amazon-mq-active-mq-5.15.16.xsd.zip

これらのスキーマは、設定ファイルの検証とサニタイズに使用できます。Amazon MQ では、XML ファイルをアップロードして設定を提供することもできます。XML ファイルをアップロードすると、Amazon MQ は、スキーマに従って無効および禁止されている設定パラメータを自動的にサニタイズし、削除します。

> **Note**
>
> 属性には静的な値のみを使用できます。Amazon MQ は、Spring 式、変数、および要素参照が含まれる要素と属性を設定からサニタイズします。

Amazon MQ for ActiveMQ ブローカー設定の作成

設定には、ActiveMQ ブローカーのすべての設定が XML 形式で含まれています (ActiveMQ の `activemq.xml` ファイルに似ています)。設定は、ブローカーを作成する前に作成することができます。作成後、設定を 1 つ、または複数のブローカーに適用できます。設定は直ちに適用する、またはメンテナンスウィンドウ中に適用することができます。

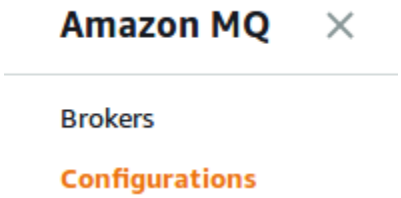
以下の例では、 AWS マネジメントコンソールを使用して Amazon MQ ブローカーの設定を作成し、適用する方法を説明します。

> **Important**
>
> 設定は `DeleteConfiguration` API を使用してのみ削除できます。詳細については、「Amazon MQ API リファレンス」の「設定」を参照してください。

新しい設定の作成

新しいブローカー設定を作成するには、まず新しい設定を作成します。

1. Amazon MQ コンソールにサインインします。
2. 左側のナビゲーションパネルを展開し、[設定] を選択します。



3. [設定] ページで、[Create configuration (設定の作成)] を選択します。
4. [設定の作成] ページの [詳細] セクションで [設定名] (`MyConfiguration` など) を入力し、[ブローカーエンジン] のバージョンを選択します。

> **Note**
>
> Amazon MQ for ActiveMQ がサポートする ActiveMQ エンジンバージョンの詳細については、「the section called “バージョン管理”」を参照してください。

5. [Create configuration] (設定の作成) をクリックします。

新しい設定リビジョンの作成

ブローカー設定を作成したら、設定リビジョンを使用して設定を編集する必要があります。

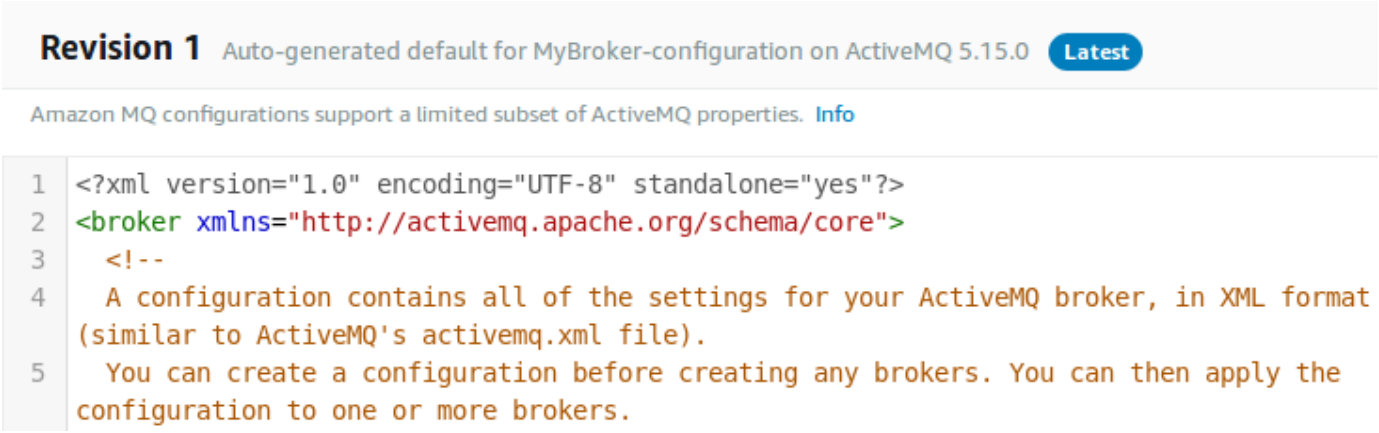
1. 設定リストから、[***MyConfiguration***] を選択します。

> **Note**
>
> 設定の最初のリビジョンは常に、Amazon MQ が設定を作成するときに作成されます。

[***MyConfiguration***] ページに、新しい設定リビジョンで使用されるブローカーのエンジンタイプとバージョン (Apache ActiveMQ 5.15.16 など) が表示されます。

2. [Configuration details] タブに、設定リビジョン番号、説明、およびブローカー設定が XML 形式で表示されます。

> **Note**
>
> 現在の設定を編集すると、設定の新しいリビジョンが作成されます。



3. [Edit configuration] (設定の編集) をクリックして、XML 設定を変更します。
4. [Save] (保存) をクリックします。

   [Save revision] (リビジョンの保存) ダイアログボックスが表示されます。
5. (オプション) `A description of the changes in this revision` を入力します。
6. [保存] を選択します。

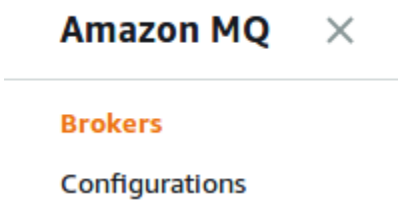
   設定の新しいリビジョンが保存されます。

> **Important**
>
> Amazon MQ コンソールは、スキーマに従って、無効および禁止されている設定パラメータを自動的にサニタイズします。許可されている XML パラメータの詳細および完全なリストについては、「Amazon MQ Broker Configuration Parameters」を参照してください。

設定リビジョンをブローカーに適用する

設定を変更したら、設定リビジョンをブローカーに適用できます。

1. 左側のナビゲーションパネルを展開し、[Brokers (ブローカー)] を選択します。



2. ブローカーリストからブローカーを選択して (MyBroker など)、[Edit] (編集) をクリックします。
3. [Edit ***MyBroker***] (MyBroker の編集) ページの [Configuration] (設定) セクションで [Configuration] (設定) と [Revision] (リビジョン) を選択してから、[Schedule Modifications] (変更をスケジュールする) をクリックします。
4. [ブローカー変更のスケジュール] セクションで、変更を [次回のスケジュールされたメンテナンスウィンドウ中] に適用するか、[即時] 適用するかを選択します。

> **Important**
>
> 再起動中、シングルインスタンスブローカーはオフラインになります。クラスターブローカーの場合、ブローカーの再起動中にダウンするノードは 1 つだけです。

5. [Apply] (適用) をクリックします。

   設定リビジョンが指定された時刻にブローカーに適用されます。

Amazon MQ for ActiveMQ 設定リビジョンの編集

ブローカーに設定リビジョンを適用した後で、そのリビジョンを編集する必要が生じることがあります。設定リビジョンを編集するには、次の手順に従います。

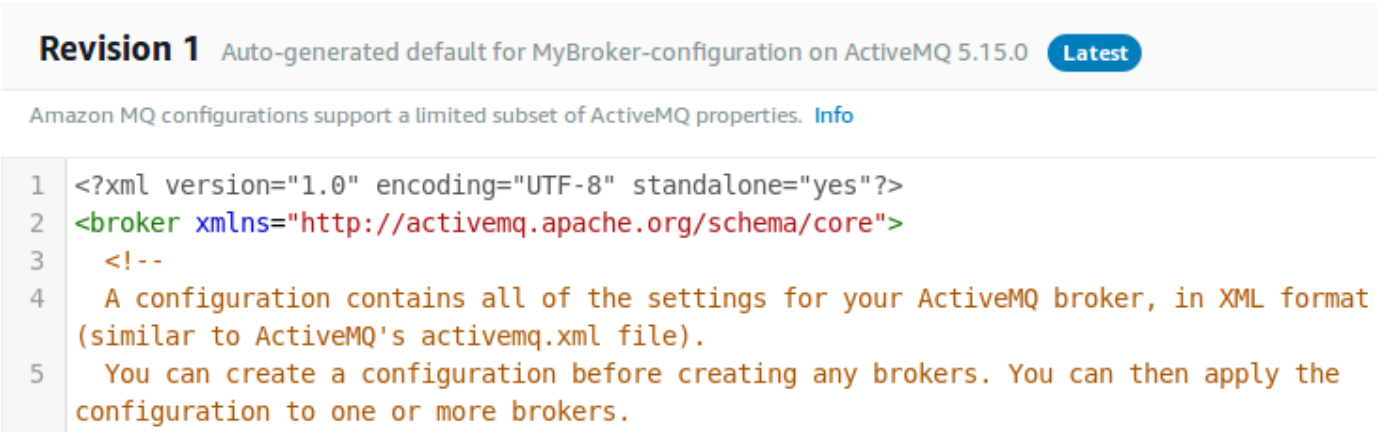
1. Amazon MQ コンソールにサインインします。
2. ブローカーリストからブローカーを選択して (MyBroker など)、[Edit] (編集) をクリックします。
3. [***MyBroker***] ページで、[編集] を選択します。
4. [Edit ***MyBroker***] (MyBroker の編集) ページの [Configuration] (設定) セクションで [Configuration] (設定) と [Revision] (リビジョン) を選択してから、[Edit] (変更) をクリックします。

> **Note**
>
> ブローカーの作成時に設定を選択する場合を除き、最初のリビジョンは、常に Amazon MQ がブローカーを作成する時に作成されます。

[***MyBroker***] ページに、設定が使用するブローカーのエンジンタイプとバージョン (Apache ActiveMQ 5.15.8 など) が表示されます。

5. [Configuration details] タブに、設定リビジョン番号、説明、およびブローカー設定が XML 形式で表示されます。

> **Note**
>
> 現在の設定を編集すると、設定の新しいリビジョンが作成されます。



6. [Edit configuration] (設定の編集) をクリックして、XML 設定を変更します。
7. [Save] (保存) をクリックします。

   [Save revision] (リビジョンの保存) ダイアログボックスが表示されます。
8. (オプション) `A description of the changes in this revision` を入力します。
9. [保存] を選択します。

   設定の新しいリビジョンが保存されます。

> **Important**
>
> Amazon MQ コンソールは、スキーマに従って、無効および禁止されている設定パラメータを自動的にサニタイズします。許可されている XML パラメータの詳細および完全なリストについては、「Amazon MQ Broker Configuration Parameters」を参照してください。

Amazon MQ の設定で許可されている要素

以下は、Amazon MQ 設定で許可されている要素の詳しいリストです。詳細については、Apache ActiveMQ ドキュメントの XML 設定を参照してください。

Element
abortSlowAckConsumerStrategy (属性)
abortSlowConsumerStrategy (属性)
authorizationEntry (属性)
authorizationMap (子コレクション要素)
authorizationPlugin (子コレクション要素)
broker (属性 | 子コレクション要素)
cachedMessageGroupMapFactory (属性)
compositeQueue (属性 | 子コレクション要素)
compositeTopic (属性 | 子コレクション要素)
constantPendingMessageLimitStrategy (属性)
discarding (属性)
discardingDLQBrokerPlugin (属性)
fileCursor
fileDurableSubscriberCursor
fileQueueCursor
filteredDestination (属性)
fixedCountSubscriptionRecoveryPolicy (属性)

Element
fixedSizedSubscriptionRecoveryPolicy (属性)
forcePersistencyModeBrokerPlugin (属性)
individualDeadLetterStrategy (属性)
lastImageSubscriptionRecoveryPolicy
messageGroupHashBucketFactory (属性)
mirroredQueue (属性)
noSubscriptionRecoveryPolicy
oldestMessageEvictionStrategy (属性)
oldestMessageWithLowestPriorityEvictionStrategy (属性)
policyEntry (属性 | 子コレクション要素)
policyMap (子コレクション要素)
prefetchRatePendingMessageLimitStrategy (属性)
priorityDispatchPolicy
priorityNetworkDispatchPolicy
queryBasedSubscriptionRecoveryPolicy (属性)
queue (属性)
redeliveryPlugin (属性 | 子コレクション要素)
redeliveryPolicy (属性)
redeliveryPolicyMap (子コレクション要素)
retainedMessageSubscriptionRecoveryPolicy (子コレクション要素)

Element
roundRobinDispatchPolicy
sharedDeadLetterStrategy (属性 | 子コレクション要素)
simpleDispatchPolicy
simpleMessageGroupMapFactory
statisticsBrokerPlugin
storeCursor
storeDurableSubscriberCursor (属性)
strictOrderDispatchPolicy
tempDestinationAuthorizationEntry (属性)
tempQueue (属性)
tempTopic (属性)
timedSubscriptionRecoveryPolicy (属性)
timeStampingBrokerPlugin (属性)
topic (属性)
transportConnector (属性)
uniquePropertyMessageEvictionStrategy (属性)
virtualDestinationInterceptor (子コレクション要素)
virtualTopic (属性)
vmCursor
vmDurableCursor

Element
vmQueueCursor

Amazon MQ 設定で許可されている要素とその属性

以下は、Amazon MQ 設定で許可されている要素とその属性の詳しいリストです。詳細については、Apache ActiveMQ ドキュメントの XML 設定を参照してください。

Element	属性
abortSlowAckConsumerStrategy	abortConnection
	checkPeriod
	ignoreIdleConsumers
	ignoreNetworkConsumers
	maxSlowCount
	maxSlowDuration
	maxTimeSinceLastAck
	name
abortSlowConsumerStrategy	abortConnection
	checkPeriod
	ignoreNetworkConsumers
	maxSlowCount
	maxSlowDuration
	name
authorizationEntry	admin

Element	属性
	queue
	read
	tempQueue
	tempTopic
	topic
	write
broker	advisorySupport
	allowTempAutoCreationOnSend
	cacheTempDestinations
	consumerSystemUsagePortion
	dedicatedTaskRunner
	deleteAllMessagesOnStartup
	keepDurableSubsActive
	enableMessageExpirationOnActiveDurableSubs
	maxPurgedDestinationsPerSweep
	maxSchedulerRepeatAllowed
	monitorConnectionSplits
	networkConnectorStartAsync
	offlineDurableSubscriberTaskSchedule

Element	属性
	offlineDurableSubscriberTimeout
	persistenceThreadPriority
	persistent
	populateJMSXUserID
	producerSystemUsagePortion
	rejectDurableConsumers
	rollbackOnlyOnAsyncException
	schedulePeriodForDestinatio nPurge
	schedulerSupport
	splitSystemUsageForProducer sConsumers
	taskRunnerPriority
	timeBeforePurgeTempDestinations
	useAuthenticatedPrincipalFo rJMSXUserID
	useMirroredQueues
	useTempMirroredQueues
	useVirtualDestSubs
	useVirtualDestSubsOnCreation
	useVirtualTopics

Element	属性
cachedMessageGroupMapFactory	cacheSize
compositeQueue	concurrentSend
	copyMessage
	forwardOnly
	name
	sendWhenNotMatched
compositeTopic	concurrentSend
	copyMessage
	forwardOnly
	name
	sendWhenNotMatched
conditionalNetworkBridgeFilterFactory	rateDuration
	rateLimit
	replayDelay
	replayWhenNoConsumers
	selectorAware 以下でサポート Apache ActiveMQ 5.16.x
constantPendingMessageLimitStrategy	limit

Element	属性
discarding	deadLetterQueue
	enableAudit
	expiration
	maxAuditDepth
	maxProducersToAudit
	processExpired
	processNonPersistent
discardingDLQBrokerPlugin	dropAll
	dropOnly
	dropTemporaryQueues
	dropTemporaryTopics
	reportInterval
filteredDestination	queue
	selector
	topic
fixedCountSubscriptionRecoveryPolicy	maximumSize
fixedSizedSubscriptionRecoveryPolicy	maximumSize
	useSharedBuffer
forcePersistencyModeBrokerPlugin	persistenceFlag
individualDeadLetterStrategy	destinationPerDurableSubscriber

Element	属性
	enableAudit
	expiration
	maxAuditDepth
	maxProducersToAudit
	processExpired
	processNonPersistent
	queuePrefix
	queueSuffix
	topicPrefix
	topicSuffix
	useQueueForQueueMessages
	useQueueForTopicMessages
messageGroupHashBucketFactory	bucketCount
	cacheSize
mirroredQueue	copyMessage
	postfix
	prefix
oldestMessageEvictionStrategy	evictExpiredMessagesHighWatermark
oldestMessageWithLowestPriorityEvictionStrategy	evictExpiredMessagesHighWatermark

Element	属性
policyEntry	advisoryForConsumed
	advisoryForDelivery
	advisoryForDiscardingMessages
	advisoryForFastProducers
	advisoryForSlowConsumers
	advisoryWhenFull
	allConsumersExclusiveByDefault
	alwaysRetroactive
	blockedProducerWarningInterval
	consumersBeforeDispatchStarts
	cursorMemoryHighWaterMark
	doOptimzeMessageStorage
	durableTopicPrefetch
	enableAudit
	expireMessagesPeriod
	gcInactiveDestinations
	gcWithNetworkConsumers
	inactiveTimeoutBeforeGC
	inactiveTimoutBeforeGC
	includeBodyForAdvisory

Element	属性
	lazyDispatch
	maxAuditDepth
	maxBrowsePageSize
	maxDestinations
	maxExpirePageSize
	maxPageSize
	maxProducersToAudit
	maxQueueAuditDepth
	memoryLimit
	messageGroupMapFactoryType
	minimumMessageSize
	optimizedDispatch
	optimizeMessageStoreInFligh tLimit
	persistJMSRedelivered
	prioritizedMessages
	producerFlowControl
	queue
	queueBrowserPrefetch
	queuePrefetch
	reduceMemoryFootprint

Element	属性
	sendAdvisoryIfNoConsumers
	sendFailIfNoSpace
	sendFailIfNoSpaceAfterTimeout 以下でサポート Apache ActiveMQ 5.16.4 以上
	sendDuplicateFromStoreToDLQ
	storeUsageHighWaterMark
	strictOrderDispatch
	tempQueue
	tempTopic
	timeBeforeDispatchStarts
	topic
	topicPrefetch
	useCache
	useConsumerPriority
usePrefetchExtension	
prefetchRatePendingMessageLimitStrategy	multiplier
queryBasedSubscriptionRecoveryPolicy	query

Element	属性
queue	DLQ
	physicalName
redeliveryPlugin	fallbackToDeadLetter
	sendToDlqIfMaxRetriesExceeded
redeliveryPolicy	backOffMultiplier
	collisionAvoidancePercent
	initialRedeliveryDelay
	maximumRedeliveries
	maximumRedeliveryDelay
	preDispatchCheck
	queue
	redeliveryDelay
	tempQueue
	tempTopic
	topic
	useCollisionAvoidance
	useExponentialBackOff
sharedDeadLetterStrategy	enableAudit
	expiration
	maxAuditDepth

Element	属性
storeDurableSubscriberCursor	maxProducersToAudit
	processExpired
	processNonPersistent
	immediatePriorityDispatch
	useCache
tempDestinationAuthorizationEntry	admin
	queue
	read
	tempQueue
	tempTopic
	topic
	write
tempQueue	DLQ
	physicalName
tempTopic	DLQ
	physicalName
timedSubscriptionRecoveryPolicy	zeroExpirationOverride
timeStampingBrokerPlugin	recoverDuration
	futureOnly
	processNetworkMessages

Element	属性
	ttlCeiling
topic	DLQ
	physicalName
transportConnector	name
	updateClusterClients
	rebalanceClusterClients
	updateClusterClientsOnRemove
uniquePropertyMessageEvictionStrategy	evictExpiredMessagesHighWatermark
	propertyName
virtualTopic	concurrentSend
	local
	dropOnResourceLimit
	name
	postfix
	prefix
	selectorAware
	setOriginalDestination
	transactedSend

Amazon MQ 親要素属性

以下は、親要素属性の詳しい説明です。詳細については、Apache ActiveMQ ドキュメントの XML 設定を参照してください。

トピック

- ブローカー

ブローカー

`broker` は親コレクションの要素です。

属性

networkConnectionStartAsync

ネットワークのレイテンシーを短縮し、他のネットワークをタイムリーに起動できるようにするには、`<networkConnectionStartAsync>` タグを使用します。このタグは、ブローカーの起動とは非同期に、エグゼキューターを使用してネットワーク接続を並列に起動するようにブローカーに指示します。

デフォルト: `false`

サンプル設定

```
<broker networkConnectorStartAsync="false"/>
```

Amazon MQ 設定で許可されている要素、子コレクション要素、およびそれらの子要素

以下は、Amazon MQ 設定で許可されている要素、子コレクション要素、およびそれらの子要素の詳しいリストです。詳細については、Apache ActiveMQ ドキュメントの XML 設定を参照してください。

Element	子コレクション要素	子要素
`authorizationMap`	`authorizationEntries`	`authorizationEntry`
		`tempDestinationAuthorizationEntry`

Element	子コレクション要素	子要素
	defaultEntry	authorizationEntry
		tempDestinationAuthorizationEntry
	tempDestinationAuthorizationEntry	tempDestinationAuthorizationEntry
authorizationPlugin	map	authorizationMap
broker	destinationInterceptors	mirroredQueue
		virtualDestinationInterceptor
	destinationPolicy	policyMap
	destinations	queue
		tempQueue
		tempTopic
		topic
	networkConnectors	networkConnector
	persistenceAdapter	kahaDB
	plugins	authorizationPlugin
		discardingDLQBrokerPlugin
		forcePersistencyModeBrokerPlugin
		redeliveryPlugin

Element	子コレクション要素	子要素
		statisticsBrokerPlugin
		timeStampingBrokerPlugin
	systemUsage	systemUsage
	transportConnector	name
		updateClusterClients
		rebalanceClusterClients
		updateClusterClientsOnRemove
compositeQueue	forwardTo	queue
		tempQueue
		tempTopic
		topic
		filteredDestination
compositeTopic	forwardTo	queue
		tempQueue
		tempTopic
		topic
		filteredDestination
policyEntry	deadLetterStrategy	discarding

Element	子コレクション要素	子要素
		individualDeadLetterStrategy
		sharedDeadLetterStrategy
	destination	queue
		tempQueue
		tempTopic
		topic
	dispatchPolicy	priorityDispatchPolicy
		priorityNetworkDispatchPolicy
		roundRobinDispatchPolicy
		simpleDispatchPolicy
		strictOrderDispatchPolicy
		clientIdFilterDispatchPolicy
	messageEvictionStrategy	oldestMessageEvictionStrategy
		oldestMessageWithLowestPriorityEvictionStrategy

Element	子コレクション要素	子要素
		uniquePropertyMessageEvictionStrategy
	messageGroupMapFactory	cachedMessageGroupMapFactory
		messageGroupHashBucketFactory
		simpleMessageGroupMapFactory
	pendingDurableSubscriberPolicy	fileDurableSubscriberCursor
		storeDurableSubscriberCursor
		vmDurableCursor
	pendingMessageLimitStrategy	constantPendingMessageLimitStrategy
		prefetchRatePendingMessageLimitStrategy
	pendingQueuePolicy	fileQueueCursor
		storeCursor
		vmQueueCursor
	pendingSubscriberPolicy	fileCursor
		vmCursor

Element	子コレクション要素	子要素
	slowConsumerStrategy	abortSlowAckConsumerStrategy
		abortSlowConsumerStrategy
	subscriptionRecoveryPolicy	fixedCountSubscriptionRecoveryPolicy
		fixedSizedSubscriptionRecoveryPolicy
		lastImageSubscriptionRecoveryPolicy
		noSubscriptionRecoveryPolicy
		queryBasedSubscriptionRecoveryPolicy
		retainedMessageSubscriptionRecoveryPolicy
timedSubscriptionRecoveryPolicy		
policyMap	defaultEntry	policyEntry
	policyEntries	policyEntry
redeliveryPlugin	redeliveryPolicyMap	redeliveryPolicyMap
redeliveryPolicyMap	defaultEntry	redeliveryPolicy
	redeliveryPolicyEntries	redeliveryPolicy

Element	子コレクション要素	子要素
retainedMessageSubscriptionRecoveryPolicy	wrapped	fixedCountSubscriptionRecoveryPolicy
		fixedSizedSubscriptionRecoveryPolicy
		lastImageSubscriptionRecoveryPolicy
		noSubscriptionRecoveryPolicy
		queryBasedSubscriptionRecoveryPolicy
		retainedMessageSubscriptionRecoveryPolicy
		timedSubscriptionRecoveryPolicy
sharedDeadLetterStrategy	deadLetterQueue	queue
		tempQueue
		tempTopic
		topic
virtualDestinationInterceptor	virtualDestinations	compositeQueue
		compositeTopic
		virtualTopic

Amazon MQ 子要素属性

以下は、子要素属性の詳しい説明です。詳細については、Apache ActiveMQ ドキュメントの XML 設定を参照してください。

トピック

authorizationEntry

`authorizationEntry` は `authorizationEntries` 子コレクション要素の子です。

属性

管理|読み取り|書き込み

ユーザーのグループに付与されているアクセス許可。詳細については、「認可マップを常に設定する」を参照してください。

`activemq-webconsole` グループが含まれない認可マップを指定する場合、Amazon MQ ブローカーにメッセージを送信する権限、またはブローカーからメッセージを受信する権限がグループにないことから、ActiveMQ ウェブコンソールは使用できません。

デフォルト: `null`

サンプル設定

```
<authorizationPlugin>
                                <map>
                                    <authorizationMap>
                                        <authorizationEntries>
                                            <authorizationEntry admin="admins,activemq-
webconsole" read="admins,users,activemq-webconsole" write="admins,activemq-webconsole"
 queue=">"/>
                                            <authorizationEntry admin="admins,activemq-
webconsole" read="admins,users,activemq-webconsole" write="admins,activemq-webconsole"
 topic=">"/>
```

```
                                    </authorizationEntries>
                                </authorizationMap>
                            </map>
                            </authorizationPlugin>
```

> **Note**
>
> Amazon MQ 上の ActiveMQ の `activemq-webconsole` グループには、すべてのキューとトピックに対する管理者アクセス許可があります。このグループのすべてのユーザーは管理者アクセス権を持ちます。

networkConnector

`networkConnector` は `networkConnectors` 子コレクション要素の子です。

トピック

- 属性
- 構成例

属性

conduitSubscriptions

ブローカーのネットワークのネットワーク接続が、同じ送信先にサブスクライブしている複数のコンシューマーを 1 つのコンシューマーとして扱うかどうかを指定します。たとえば、`conduitSubscriptions` が `true` に設定されていて、2 つのコンシューマーがブローカー B に接続して送信先から消費する場合、ブローカー B は、ブローカー A へのネットワーク接続を介してサブスクリプションを単一の論理サブスクリプションに結合するので、メッセージの単一コピーのみがブローカー A からブローカー B に転送されます。

> **Note**
>
> `conduitSubscriptions` を `true` に設定すると、冗長なネットワークトラフィックを減らすことができます。ただし、この属性を使用すると、コンシューマー間でのメッセージのロードバランシングに影響が出る可能性があり、特定のシナリオ (JMS メッセージセレクタや耐久性のあるトピックなど) では正しくない動作を引き起こす可能性があります。

デフォルト: `true`

二重

ブローカーのネットワーク内の接続を使用し、またメッセージを生成するかどうかを指定します。ブ たとえば、ブローカー A が非二重モードでブローカー B への接続を作成した場合、メッセージはブローカー A からブローカー B にのみ転送できます。ただし、ブローカー A がブローカー B への二重接続を作成した場合、ブローカー B は <networkConnector> を設定しなくてもメッセージをブローカー A に転送できます。

デフォルト: `false`

名前

ブローカーのネットワークのブリッジの名前。

デフォルト: `bridge`

uri

ブローカーのネットワークの 2 つのブローカーのうちの 1 つ (または複数のブローカー) のワイヤレベルプロトコルエンドポイント。

デフォルト: `null`

username

ブローカーのネットワークのブローカーに共通のユーザー名。

デフォルト: `null`

構成例

> **Note**
>
> `networkConnector` を使用してブローカーのネットワークを定義するときは、ブローカーに共通のユーザーのパスワードを含めないでください。

2 つのブローカーとブローカーのネットワーク

この設定では、2 つのブローカーがブローカーのネットワークで接続されています。ネットワークコネクターの名前は `connector_1_to_2`、ブローカーに共通のユーザー名は `myCommonUser`、接続

は duplex、そして OpenWire エンドポイント URI は static: というプレフィックスは、ブローカー間の 1 対 1 の接続を示します。

```
<networkConnectors>
                                    <networkConnector name="connector_1_to_2"
 userName="myCommonUser" duplex="true"
                                        uri="static:(ssl://
b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-east-2.amazonaws.com:61617)"/>
                                    </networkConnectors>
```

詳細については、「Configure Network Connectors for Your Broker」を参照してください。

複数のブローカーのあるブローカーのネットワーク

この設定では、複数のブローカーがブローカーのネットワークで接続されています。ネットワークコネクターの名前は connector_1_to_2、ブローカーに共通のユーザー名は myCommonUser、接続は duplex です。OpenWire エンドポイント URI のカンマ区切りのリストの前には masterslave: というプレフィックスが付き、ブローカー間のフェイルオーバー接続を示します。ブローカーからブローカーへのフェイルオーバーはランダム化されず、再接続の試行は無期限に続きます。

```
<networkConnectors>
                                    <networkConnector name="connector_1_to_2"
 userName="myCommonUser" duplex="true"
                                        uri="masterslave:(ssl://
b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-east-2.amazonaws.com:61617,
                                        ssl://
b-9876l5k4-32ji-109h-8gfe-7d65c4b132a1-2.mq.us-west-2.amazonaws.com:61617)"/>
                                    </networkConnectors>
```

> **Note**
>
> ブローカーのネットワークの masterslave: プレフィックスを使用することをお勧めします。プレフィックスはより明示的な static:failover:()?randomize=false&maxReconnectAttempts=0 構文と完全に一致します。

> **Note**
>
> この XML 設定ではスペースを使用できません。

kahaDB

`kahaDB` は `persistenceAdapter` 子コレクション要素の子です。

属性

concurrentStoreAndDispatchQueues

キューの同時保存とディスパッチを使用するかどうかを指定します。詳細については、「低速コンシューマーのキューに対して同時保存とディスパッチを無効にする」を参照してください。

デフォルト: `true`

cleanupOnStop

> **以下でサポート**
>
> Apache ActiveMQ 15.16.x 以上

無効にされていると、ブローカーが停止されたときにガベージコレクションおよびクリーンアップが実行されず、シャットダウンプロセスの速度が上がります。高速化は、大規模なデータベースやスケジューラデータベースの場合に有用です。

デフォルト: `true`

journalDiskSyncInterval

`journalDiskSyncStrategy=periodic` の場合にディスク同期を実行する間隔 (ミリ秒)。詳細については、Apache ActiveMQ kahaDB のドキュメントを参照してください。

デフォルト: `1000`

journalDiskSyncStrategy

> **以下でサポート**
>
> Apache ActiveMQ 15.14.x 以上

ディスク同期ポリシーを設定します。詳細については、Apache ActiveMQ kahaDB のドキュメントを参照してください。

デフォルト: `always`

> **Note**
>
> ActiveMQ のドキュメントでは、データ損失は `journalDiskSyncInterval` の長さに制限されており、デフォルトは 1 秒です。厳密には言えませんが、データ損失はこの間隔よりも長くなる可能性があります。注意してください。

preallocationStrategy

新しいジャーナルファイルが必要になったときにブローカーがジャーナルファイルの事前割り当てを試みる方法を設定します。詳細については、Apache ActiveMQ kahaDB のドキュメントを参照してください。

デフォルト: `sparse_file`

サンプル設定

Example

```
<broker xmlns="http://activemq.apache.org/schema/core">
                                <persistenceAdapter>
                                <kahaDB preallocationStrategy="zeros"
 concurrentStoreAndDispatchQueues="false" journalDiskSyncInterval="10000"
 journalDiskSyncStrategy="periodic"/>
                            </persistenceAdapter>
                            </broker>
```

systemUsage

`systemUsage` は `systemUsage` 子コレクション要素の子です。プロデューサーの速度を遅くするまでにブローカーが使用する領域の最大量を制御します。詳細については、Apache ActiveMQ のドキュメントの Producer Flow Control を参照してください。

子要素

memoryUsage

`memoryUsage` は `systemUsage` 子要素の子です。メモリ使用量を管理します。本番稼働での作業セットの使用を制御できるように、`memoryUsage` を使用してメモリ使用量を追跡します。詳細については、Apache ActiveMQ のドキュメントの schema を参照してください。

子要素

memoryUsage は memoryUsage 子要素の子です。

属性

percentOfJvmHeap

0 ~ 70 の整数。

デフォルト: 70

属性

sendFailIfNoSpace

空き領域がない場合に send( ) メソッドが失敗するかどうかを設定します。デフォルト値は false で、領域が空くまで send( ) メソッドをブロックします。詳細については、Apache Active MQ のドキュメントの schema を参照してください。

デフォルト: false

sendFailIfNoSpaceAfterTimeout

デフォルト: null

サンプル設定

Example

```
<broker xmlns="http://activemq.apache.org/schema/core">
                                <systemUsage>
                                <systemUsage sendFailIfNoSpace="true"
 sendFailIfNoSpaceAfterTimeout="2000">
                                    <memoryUsage>
                                        <memoryUsage  percentOfJvmHeap="60" />
                                    </memoryUsage>>
                                </systemUsage>
                                </systemUsage>
                            </broker>
                            </persistenceAdapter>
```

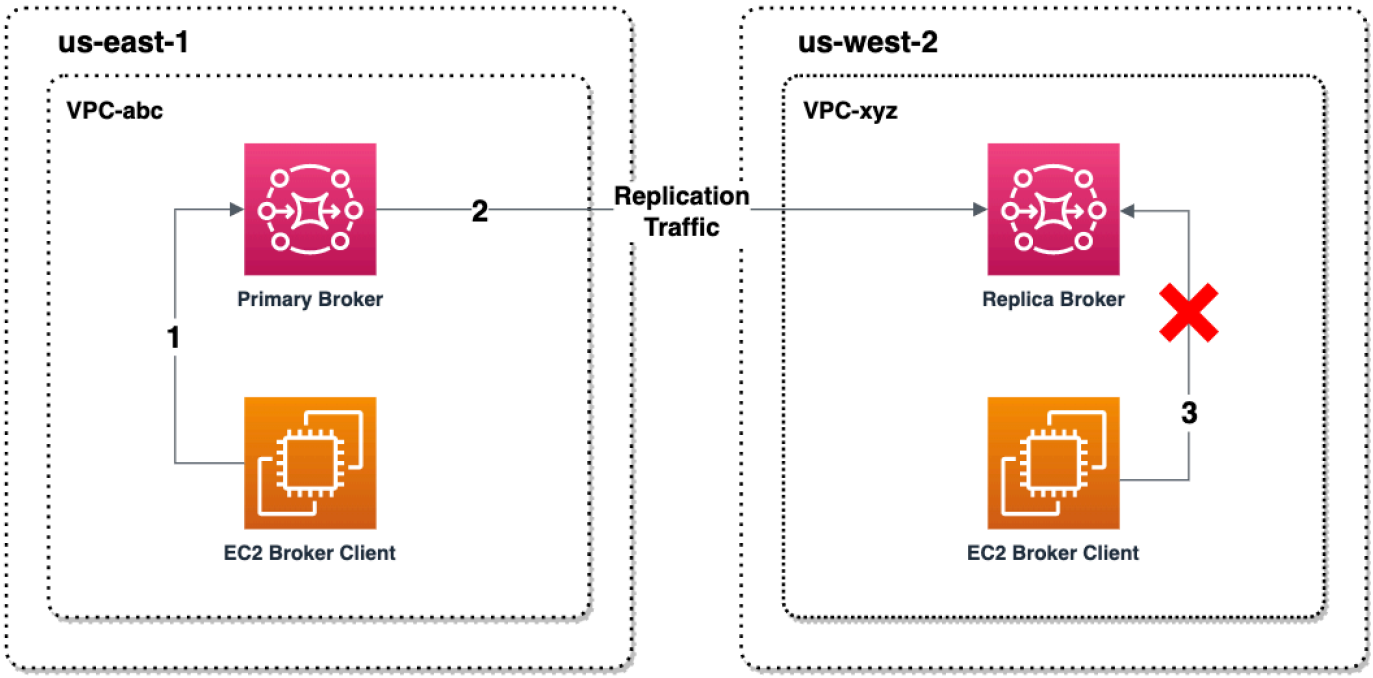
Amazon MQ for ActiveMQ のクロスリージョンデータレプリケーション

Amazon MQ for ActiveMQ には、クロスリージョンデータレプリケーション (CRDR) 機能があります。この機能では、プライマリ AWS リージョンのプライマリブローカーからレプリカリージョンのレプリカブローカーへの非同期メッセージレプリケーションが可能です。Amazon MQ API にフェイルオーバーリクエストを発行すると、現在のレプリカブローカーはプライマリブローカーのロールに昇格され、現在のプライマリブローカーはレプリカのロールに降格されます。

クロスリージョンデータレプリケーション用のプライマリブローカーとレプリカブローカー

プライマリ AWS リージョンのプライマリブローカーからレプリカリージョンのレプリカブローカーへの非同期データレプリケーション用のプライマリブローカーとレプリカブローカーを作成できます。プライマリリージョンは、プライマリブローカーと呼ばれるアクティブ/スタンバイブローカーの冗長ペアで構成されます。セカンダリリージョンは、レプリカブローカーと呼ばれるアクティブ/スタンバイブローカーの冗長ペアで構成されます。

次の図は、セカンダリリージョンのレプリカブローカーが、プライマリリージョンのプライマリブローカーから非同期にレプリケートされたデータを受信する様子を示しています。



プライマリブローカーとレプリカブローカーは、クロスリージョンのデータ復旧ソリューションとして機能します。プライマリリージョンのプライマリブローカーに障害が発生した場合、スイッチオーバーまたはフェイルオーバーを開始することで、セカンダリリージョンのレプリカブローカーをプライマリに昇格させることができます。元のプライマリブローカーはレプリカブローカーになり、元のレプリカブローカーはプライマリブローカーに昇格されます。プライマリブローカーとレプリカブローカーの作成手順については、「Amazon MQ クロスリージョンデータレプリケーションブローカーの作成」を参照してください。

> **Note**
>
> アクティブ/スタンバイブローカーでのみ使用できます。
> ミラーキューでは使用できません。

Amazon MQ クロスリージョンデータレプリケーションブローカーの作成

クロスリージョンデータレプリケーション (CRDR) を使用すると、必要に応じて 2 つの AWS リージョンの Amazon MQ for ActiveMQ メッセージブローカーを切り替えることができます。既存のブローカーをプライマリブローカーとして指定し、このブローカーのレプリカを作成することも、新しいプライマリブローカーとレプリカブローカーを一緒に作成することもできます。その後、Amazon MQ `Promote` API オペレーションを使用して、レプリカブローカーをプライマリブローカーのロールに昇格させることができます。プライマリブローカーとレプリカブローカーの詳細については、「クロスリージョンデータレプリケーション用のプライマリブローカーとレプリカブローカー」を参照してください。

次の手順では、Amazon MQ マネジメントコンソールを使用してレプリカブローカーを作成および設定する方法について説明します。

トピック

- 前提条件
- ステップ 1 (オプション): 新しいプライマリブローカーを作成する
- ステップ 2: 既存のブローカーのレプリカを作成する

前提条件

クロスリージョンデータレプリケーション機能を使用するには、以下の前提条件を確認して遵守する必要があります。

- バージョン: クロスリージョンデータレプリケーション機能は、バージョン 5.17.6 以降の Amazon MQ for ActiveMQ ブローカーでのみ利用できます。
- リージョン: クロスリージョンデータレプリケーションは、米国東部 (オハイオ)、米国東部 (バージニア北部)、米国西部 (オレゴン)、および米国西部 (北カリフォルニア) の各リージョンでサポートされます。
- インスタンスタイプ: クロスリージョンデータレプリケーションは、`mq.m5.large` 以上のブローカーのインスタンスサイズでのみ利用できます。
- デプロイタイプ: クロスリージョンデータレプリケーションは、複数のアベイラビリティーゾーンデプロイのアクティブ/スタンバイブローカーでのみ利用できます。
- ブローカーのステータス: ブローカーステータスが `Running` のプライマリブローカーのレプリカブローカーのみを作成できます。

ステップ 1 (オプション): 新しいプライマリブローカーを作成する

新しいプライマリブローカーを作成する

1. Amazon MQ コンソールにサインインします。
2. Amazon MQ コンソールの [ブローカー] ページで、[ブローカーの作成] を選択します。
3. [Select broker engine] (ブローカーエンジンの選択) ページで [Apache ActiveMQ] を選択します。
4. [Select deployment and storage] (デプロイとストレージタイプの選択) ページの [Deployment mode and storage type] (デプロイモードとストレージタイプ) セクションで、以下を実行します。

    - [デプロイモード] で、[アクティブ/スタンバイブローカー] を選択します。アクティブ/スタンバイブローカーは、2 つの異なるアベイラビリティーゾーンで冗長ペアとして設定された 2 つのブローカーで構成されます。これらのブローカーは、アプリケーションおよび Amazon EFS と同期的に通信します。詳細については、「Amazon MQ for ActiveMQ ブローカーのデプロイオプション」を参照してください。
5. [Next] (次へ) を選択します。
6. [Configure settings] (設定の定義) ページの [Details] (詳細) セクションで、以下を実行します。

    a. [Broker name] (ブローカー名) を入力します。

> **Important**
>
> 個人を特定できる情報 (PII) などの機密情報や秘匿性の高い情報はタグに追加しないでください。ブローカー名は、CloudWatch Logs を含む他の AWS サービスからアクセスできます。ブローカー名は、プライベートデータや機密データとして使用することを意図していません。

b. [Broker instance type] (ブローカーインスタンスタイプ) を選択します (mq.m5.large など)。詳細については、「Broker instance types」を参照してください。

7. [ActiveMQ Web Console access] (ActiveMQ ウェブコンソールアクセス) セクションで、[Username] (ユーザーネーム) と [Password] (パスワード) を入力します。ブローカーのユーザー名とパスワードには、以下の制限が適用されます:

   - ユーザーネームに使用できるのは、英数字、ダッシュ、ピリオド、アンダースコア、およびチルデ (- . _ ~) のみです。
   - パスワードは 12 文字以上の長さで、一意の文字を少なくとも 4 つ含める必要があり、カンマ、コロン、または等号 (,:=) は使用できません。

> **Important**
>
> 個人を特定できる情報 (PII) などの機密情報や秘匿性の高い情報はブローカーのユーザー名に追加しないでください。ブローカーのユーザー名は、CloudWatch Logs を含む他の AWS サービスからアクセスできます。ブローカーのユーザー名は、プライベートデータや機密データとして使用することを意図していません。

ページ上部の緑色のフラッシュバーは、Amazon MQ がリカバリリージョンにレプリカブローカーを作成していることを示しています。ブローカーの CRDR ロールと RPO ステータスも確認できます。[CRDR ロール] 列と [RPO ステータス] 列をオフにするには、[ブローカー] テーブルの右上隅にある歯車アイコンを選択します。次に、[設定] ページで [CRDR ロール] または [RPO ステータス] をオフにします。

ステップ 2: 既存のブローカーのレプリカを作成する

1. Amazon MQ コンソールの [ブローカー] ページで、[レプリカブローカーを作成] を選択します。

2. [プライマリブローカーを選択] ページで、CRDR プライマリブローカーとして使用する既存のブローカーを選択します。次に、[次へ] を選択します。
3. [レプリカブローカーを設定] ページで、ドロップダウンメニューを使用してレプリカリージョンを選択します。
4. [レプリカブローカーのActiveMQ コンソールユーザー] セクションで、レプリカブローカーのコンソールユーザーのユーザー名とパスワードを指定します。ブローカーのユーザー名とパスワードには、以下の制限が適用されます:

   - ユーザーネームに使用できるのは、英数字、ダッシュ、ピリオド、アンダースコア、およびチルデ (- . _ ~) のみです。
   - パスワードは 12 文字以上の長さで、一意の文字を少なくとも 4 つ含める必要があり、カンマ、コロン、または等号 (,:=) は使用できません。

   > **Important**
   >
   > 個人を特定できる情報 (PII) などの機密情報や秘匿性の高い情報はブローカーのユーザー名に追加しないでください。ブローカーのユーザー名は、CloudWatch Logs を含む他の AWS サービスからアクセスできます。ブローカーのユーザー名は、プライベートデータや機密データとして使用することを意図していません。

5. [ブローカー間のアクセスをブリッジするデータレプリケーションユーザー] セクションで、プライマリブローカーとレプリカブローカーの両方にアクセスするユーザーのユーザー名とパスワードを入力します。ブローカーのユーザー名とパスワードには、以下の制限が適用されます:

   - ユーザーネームに使用できるのは、英数字、ダッシュ、ピリオド、アンダースコア、およびチルデ (- . _ ~) のみです。
   - パスワードは 12 文字以上の長さで、一意の文字を少なくとも 4 つ含める必要があり、カンマ、コロン、または等号 (,:=) は使用できません。

   > **Important**
   >
   > 個人を特定できる情報 (PII) などの機密情報や秘匿性の高い情報はブローカーのユーザー名に追加しないでください。ブローカーのユーザー名は、CloudWatch Logs を含む他の AWS サービスからアクセスできます。ブローカーのユーザー名は、プライベートデータや機密データとして使用することを意図していません。

その他の設定を行います。次に、[次へ] を選択します。

6. [確認と作成] ページで、レプリカブローカーの詳細を確認します。次に、[レプリカブローカーを作成] を選択します。
7. 次に、プライマリブローカーを再起動します。これにより、レプリカブローカーも再起動されます。ブローカーを再起動する手順については、「Rebooting a Broker」を参照してください。

ActiveMQ ブローカーの追加設定の構成の詳細については、「開始方法: ActiveMQ ブローカーの作成と接続」を参照してください。

Amazon MQ クロスリージョンデータレプリケーションブローカーの削除

プライマリクロスリージョンデータレプリケーション (CRDR) ブローカーまたはレプリカ CRDR ブローカーを削除するには、まずブローカーのペアリングを解除してから、ブローカーを再起動する必要があります。次の手順は、AWS マネジメントコンソールを使用してブローカーのペアリングを解除して再起動する方法を示しています。

1. [ブローカー] ページで、ペアリングを解除する CRDR ブローカーを選択し、[編集] を選択します。
2. ブローカーの [編集] ページの [データレプリケーション] セクションで、[ブローカーのペアリング解除] を選択します。
3. ポップアップウィンドウに「confirm」と入力し、選択を確定します。次に、[ブローカーのペアリング解除] を選択します。
4. 次に、ペアリングされていないプライマリブローカーを再起動します。これにより、レプリカブローカーも再起動されます。ブローカーを再起動する手順については、「Rebooting a Broker」を参照してください。プライマリブローカーを再起動すると、両方のブローカーのペアリングが解除され、個別に削除できます。ブローカーを削除するには、「Deleting a broker」を参照してください。

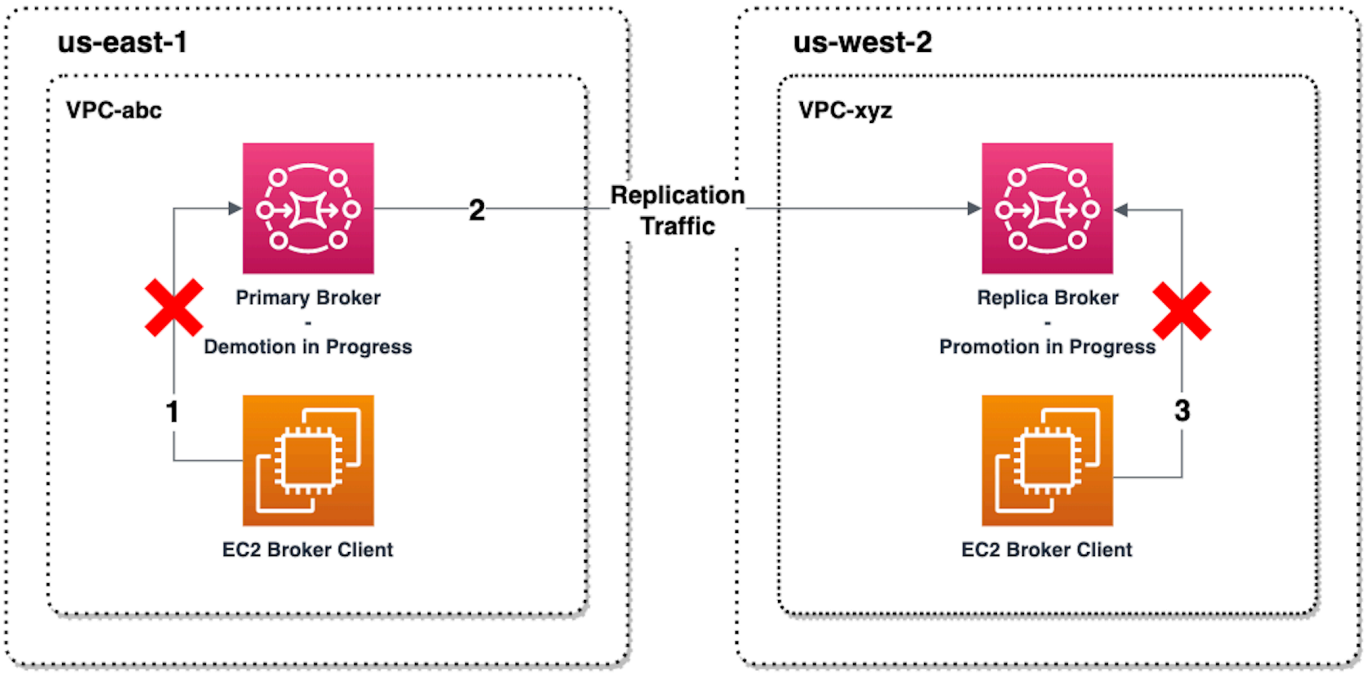
Amazon MQ レプリカブローカーをプライマリブローカーのロールに昇格させるためのスイッチオーバーまたはフェイルオーバーの開始

レプリカブローカーをプライマリブローカーのロールに昇格させる場合は、スイッチオーバーまたはフェイルオーバーを開始できます。レプリカブローカーを昇格させると、プライマリブローカーはレプリカブローカーのロールに降格されます。

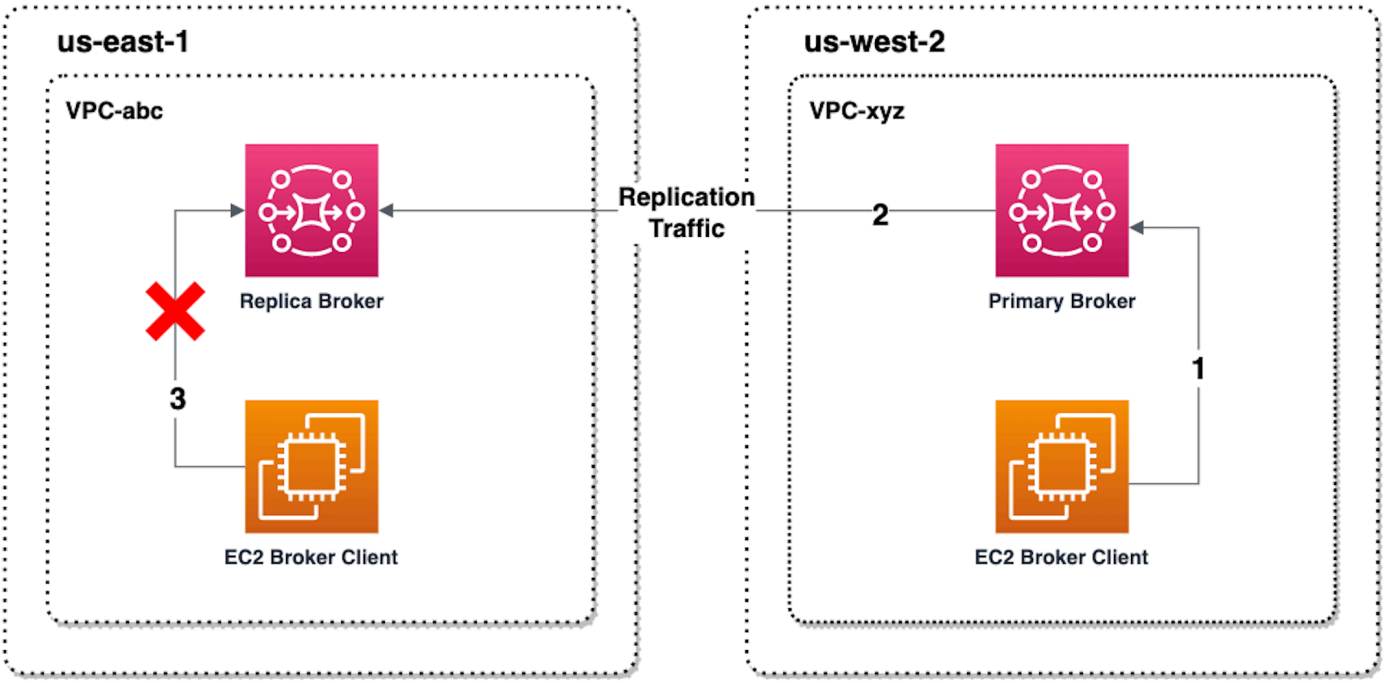
スイッチオーバーでは、可用性よりも一貫性を優先します。このフェイルオーバー操作が完了すると、ブローカーの状態が同じになることが保証されます。スイッチオーバーの場合、ブローカー間の一貫性が確立されるまでは、どちらのブローカーもクライアント接続に使用できない期間が発生する場合があります。レプリカが昇格された時点で、両方のブローカーは同じ状態になります。スイッチオーバーが成功するかどうかは、両方のリージョンの正常性とリージョン間ネットワークの成功にかかっています。

フェイルオーバーでは、一貫性よりも可用性を優先します。この操作の完了時にブローカーが同じ状態になることは保証されません。フェイルオーバーの場合、レプリケーションデータが同期されるまで、またはプライマリがシャットダウン信号を受信するまで待つことなく、レプリカブローカーがすぐにクライアントトラフィックの処理に使用可能になることが保証されます。フェイルオーバーが成功するかどうかは、元のプライマリリージョンの正常性にも、リージョン間ネットワークの成功にも依存しません。

次の図は、レプリケーションキューが空になり、ブローカーの状態が同期されるまで、どちらのブローカーもクライアント接続を受け入れないスイッチオーバーを示しています。このプロセスでは、操作が進行中で、プライマリブローカーがレプリカに降格されるまで、プライマリブローカーの VPC 内のクライアントはそれ以上の状態変更を生成できません。レプリケーションキューが空になり、2 つのブローカーが同じ状態になると、フェイルオーバー操作が完了してレプリカブローカーがプライマリに昇格されるまで、レプリカブローカーの VPC 内のクライアントはレプリカブローカーに接続できません。



次の図は、スイッチオーバープロセスが完了した後のブローカーのステータスを示しています。元のレプリカブローカーがプライマリブローカーのロールに昇格され、クライアント接続を受け入れています。クライアントはブローカーからデータを生成および利用できます。



コンソールを使用してレプリカブローカーを昇格させる

スイッチオーバーまたはフェイルオーバーを使用してレプリカブローカーを昇格させるには、Amazon MQ コンソールで次の手順に従います。

> **Note**
>
> プライマリブローカーではスイッチオーバーやフェイルオーバーを開始できません。

1. レプリカブローカーのリージョンに切り替えます。[ブローカー] テーブルで、プライマリに昇格する既存のレプリカブローカーを選択します。
2. [ブローカーの詳細] ページで、以下の操作を実行します。
    1. [レプリカを昇格させる] を選択します。
    2. ポップアップウィンドウで、[スイッチオーバー] または [フェイルオーバー] を選択します。
    3. テキストボックスに「confirm」と入力し、選択を確定します。

4. [確認] を選択してください。

フェイルオーバーを開始すると、ブローカーのステータスが [フェイルオーバー中] に変わります。フェイルオーバーが完了すると、[ブローカー] ページ上部の青い進行状況バーが緑色になります。

> **Note**
>
> 設定は、レプリカブローカーの作成時にのみレプリケートされます。それ以降の更新はレプリケートされません。

Amazon CloudWatch のクロスリージョンデータレプリケーションのメトリクス

Amazon MQ for ActiveMQ のクロスリージョンデータレプリケーション機能は、プライマリブローカーとレプリカブローカーの信頼性、可用性、パフォーマンスを維持するためのメトリクスを提供します。レプリケーションプロセス中、セカンダリリージョンのレプリカブローカーは、プライマリリージョンのプライマリブローカーから非同期でレプリケートされたデータを受信します。プライマリリージョンのプライマリブローカーに障害が発生した場合、スイッチオーバーまたはフェイルオーバーを開始することで、セカンダリリージョンのレプリカブローカーをプライマリに昇格させることができます。Amazon CloudWatch でメトリックスを表示する手順については、「Amazon MQ 向けの CloudWatch メトリクスへのアクセス」を参照してください。

CRDR のタイムスタンプ

以下のタイムスタンプは、Amazon CloudWatch でのメトリクスの計算方法を示しています。データレプリケーションプロセスには、以下の 5 つのタイムスタンプがあります。

- 現在の観測時刻 (TCO): 現在の瞬間。
- 作成時刻 (TC): プライマリブローカーがレプリケーションキューにイベントを作成した瞬間。プライマリブローカーとレプリカブローカーの両方で利用できます。
- 配信時刻 (TD): イベントがレプリカブローカーに正常に配信された瞬間。レプリカブローカーでのみ利用できます。
- 処理時刻 (TP): レプリカブローカーによってイベントが正常に処理された時刻。レプリカブローカーでのみ利用できます。
- 確認時刻 (TA): プライマリブローカーがイベントを正常に確認した瞬間。プライマリーブローカーでのみ利用できます。

CRDR CloudWatch メトリクスを使用してスイッチオーバー/フェイルオーバーのパフォーマンスを推定する

Amazon MQ は、デフォルトでブローカーのメトリクスを有効にします。Amazon CloudWatch コンソールにアクセスするか、CloudWatch API を使用して、ブローカーのメトリクスを表示できます。以下のメトリクスは、CRDR ブローカーのレプリケーションとスイッチオーバー/フェイルオーバーのパフォーマンスを理解するのに役立ちます。

Amazon MQ CloudWatch メトリクス	CRDR を使用する理由
`TotalReplicationLag`	プライマリブローカーでの最後の未確認イベントの TA から TC までの推定時間。
`ReplicationLag`	レプリカブローカーでの最後の未確認イベントの TP から TC までの推定時間。
`PrimaryWaitTime`	プライマリブローカーで最後に処理されたイベントの TCO から TC までの推定時間。
`ReplicaWaitTime`	レプリカブローカーで最後に処理されたイベントの TCO から TP までの推定時間。
`QueueSize`	プライマリブローカーのレプリケーションキューにある未確認イベントの総数。

`TotalReplicationLag` と `ReplicationLag` は、プライマリブローカーとレプリカブローカーの間の遅延レプリケーションについて説明します。この 2 つのメトリクスを使用して、進行中のスイッチオーバー操作やフェイルオーバー操作が完了するまでの時間を推定することもできます。

`PrimaryWaitTime` と `ReplicaWaitTime` は、レプリケーションプロセスで現在発生している問題を特定するために使用できます。メトリクスの値が絶えず増加している場合は、レプリケーションプロセスのパフォーマンスが低下しているか、一時停止している可能性があります。ネットワークの分

割、ブローカーの起動、長いリカバリなどの問題が原因で、レプリケーションが遅くなることがあります。

ActiveMQ チュートリアル

以下のチュートリアルでは、ActiveMQ ブローカーを作成して接続する方法を説明します。ActiveMQ Java サンプルコードを使用するには、Java Standard Edition Development Kit をインストールして、コードにいくつかの変更を行う必要があります。

トピック

- ブローカーの Amazon MQ ネットワークの作成と設定
- Amazon MQ ブローカーへの Java アプリケーションの接続
- ActiveMQ ブローカーの LDAP との統合
- ステップ 3: (オプション) AWS Lambda 関数に接続する
- ActiveMQ ブローカーユーザーの作成
- ActiveMQ ブローカーユーザーの編集
- ActiveMQ ブローカーユーザーの削除
- ActiveMQ での Java Message Service (JMS) の使用の実用例

ブローカーの Amazon MQ ネットワークの作成と設定

ブローカーのネットワークは、同時にアクティブな複数の単一インスタンスブローカー、またはアクティブ/スタンバイブローカーで構成されています。このチュートリアルでは、ソースとシンクトポロジを使用してブローカーの 2 ブローカーネットワークを作成する方法を学びます。

概念的な概要および詳細な設定情報については、以下を参照してください。

- Amazon MQ のブローカーのネットワーク
- ブローカーのネットワークを正しく設定する
- `networkConnector`
- `networkConnectionStartAsync`
- ActiveMQ ドキュメントの「ブローカーのネットワーク」

ブローカーの Amazon MQ ネットワークは、Amazon MQ コンソールを使用して作成できます。2 つのブローカーの作成を並行して開始できるため、このプロセスには約 15 分かかります。

トピック

前提条件

ブローカーのネットワークを作成するには、以下のものが必要です。

- 同時にアクティブな 2 つ以上のブローカー (このチュートリアルでは `MyBroker2` および `MyBroker1` という名前)。ブローカー作成についての詳細は、「開始方法: ActiveMQ ブローカーの作成と接続」 を参照してください。
- 2 つのブローカーは、同じ VPC またはピア接続された VPC に属している必要があります。VPC の詳細については、Amazon VPC ユーザーガイドの「Amazon VPC とは」および Amazon VPC ピアリングガイドの「VPC ピア機能とは」を参照してください。

  > **Important**
  >
  > デフォルトの VPC、サブネット、またはセキュリティグループがない場合は、それらを最初に作成する必要があります。詳細については、Amazon VPC ユーザーガイドの以下のトピックを参照してください。
  >
  > - デフォルト VPC の作成
  > - デフォルトサブネットの作成
  > - セキュリティグループを作成する

- 両方のブローカーに対して同じサインイン認証情報を持つ 2 人のユーザー。ユーザー作成の詳細については、「ActiveMQ ブローカーユーザーの作成」を参照してください。

  > **Note**
  >
  > LDAP 認証をブローカーのネットワークと統合するときは、ユーザーが ActiveMQ ブローカーと LDAP ユーザーの両方として存在することを確認してください。

以下の例では、2 つの単一インスタンスブローカーを使用します。ただし、アクティブ/スタンバイブローカー、またはブローカーデプロイモードの組み合わせを使用してブローカーのネットワークを作成できます。

ステップ 1: ブローカー間のトラフィックを許可する

ブローカーを作成した後、それらの間のトラフィックを許可する必要があります。

1. Amazon MQ コンソールの [MyBroker2] ページにある [Details] (詳細) セクションの [Security and network] (セキュリティとネットワーク) で、セキュリティグループの名前または
   
   をクリックします。

   EC2 ダッシュボードの [セキュリティグループ] ページが表示されます。
2. セキュリティグループのリストから、セキュリティグループを選択します。
3. ページ下部で、[インバウンド] を選択し、次に [編集] を選択します。
4. [インバウンドルールの編集] ダイアログボックスで、OpenWire エンドポイントのルールを追加します。

   a. [ルールの追加] を選択します。

   b. [タイプ] で、[カスタム TCP] を選択します。

   c. [ポート範囲] で、OpenWire ポートを入力します (61617)。

   d. 次のいずれかを行います。

   - 特定の IP アドレスへのアクセスを制限する場合は、[ソース] で [カスタム] を選択したままにし、`MyBroker1` の IP アドレスに続いて `/32` を入力します。(これは IP アドレスを有効な CIDR レコードに変換します)。詳細については、「Elastic Network Interfaces」を参照してください。

     > **Tip**
     >
     > `MyBroker1` の IP アドレスを取得するには、Amazon MQ コンソールでブローカーの名前を選択し、[Details] (詳細) セクションに移動します。

   - すべてのブローカーがプライベートで、同じ VPC に属している場合は、[ソース] で、[カスタム] を選択したままにし、編集しているセキュリティグループの ID を入力します。

> **Note**
>
> パブリックブローカーの場合は、IP アドレスを使用してアクセスを制限する必要があります。

e. [保存] を選択します。

これで、ブローカーはインバウンド接続を受け入れることができます。

ステップ 2: ブローカー用のネットワークコネクターを設定する

ブローカー間のトラフィックを許可すると、そのうちの 1 つのネットワーク接続を設定する必要があります。

1. ブローカー `MyBroker1` の設定リビジョンを編集します。

   a. [MyBroker1] ページで、[編集] を選択します。

   b. [MyBroker1 の編集] ページの、[設定] セクションで、[表示] を選択します。

   設定が使用するブローカーエンジンタイプとバージョン (例: [Apache ActiveMQ 5.15.0]) が表示されます。

   c. [Configuration details] タブに、設定リビジョン番号、説明、およびブローカー設定が XML 形式で表示されます。

   d. [設定の編集] を選択します。

   e. 設定ファイルの下部で、`<networkConnectors>` セクションのコメントを解除し、以下の情報を入力します。

   - ネットワークコネクターの `name`。
   - ブローカーの両方に共通の ActiveMQ ウェブコンソール`username`。
   - `duplex` 接続を有効にします。
   - 次のいずれかを行います。
     - ブローカーを単一インスタンスブローカーに接続している場合は、`MyBroker2` の `static:` プレフィックスと OpenWire エンドポイント `uri` を使用します。例:

```
<networkConnectors>
```

```
  <networkConnector name="connector_1_to_2" userName="myCommonUser"
 duplex="true"
    uri="static:(ssl://b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-
east-2.amazonaws.com:61617)"/>
</networkConnectors>
```

- ブローカーをアクティブ/スタンバイブローカーに接続している場合は、次のクエリパラメータ `?randomize=false&maxReconnectAttempts=0` を使用して、両方のブローカーの `static+failover` トランスポートと OpenWire エンドポイント `uri` を使用します。例えば、次のようになります。

```
<networkConnectors>
  <networkConnector name="connector_1_to_2" userName="myCommonUser"
 duplex="true"
    uri="static:(failover:(ssl://
b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-east-2.amazonaws.com:61617,
    ssl://b-9876l5k4-32ji-109h-8gfe-7d65c4b132a1-2.mq.us-
west-2.amazonaws.com:61617)?randomize=false&amp;maxReconnectAttempts=0)"/>
</networkConnectors>
```

> **Note**
>
> ActiveMQ ユーザーのサインイン認証情報は含めないでください。

f. [保存] を選択します。

g. [リビジョンの保存] ダイアログボックスで、「`Add network of brokers connector for MyBroker2`」と入力します。

h. [保存] を選択して設定リビジョンを保存します。

2. `MyBroker1` を編集して最新の設定リビジョンをすぐに適用するように設定します。

a. [MyBroker1] ページで、[編集] を選択します。

b. [MyBroker1 の編集] ページの、[設定] セクションで、[Schedule Modifications (スケジュールの変更)] を選択します。

c. [Schedule broker modifications (ブローカー変更のスケジュール)] セクションで、変更を適用するには、[即時] を選択します。

d. [Apply] (適用) を選択します。

`MyBroker1` が再起動され、設定リビジョンが適用されます。

ネットワークのブローカーが作成されます。

次のステップ

ブローカーのネットワークを設定したら、メッセージを作成して消費することでテストできます。

> **Important**
>
> ポート 8162 (ActiveMQ Web コンソール用) とポート 61617 (OpenWire エンドポイント用) で、ブローカー `MyBroker1` に対してローカルマシンからのインバウンド接続を有効化しておくようにしてください。
> プロデューサーとコンシューマーがブローカーのネットワークに接続できるように、セキュリティグループの設定を調整する必要がある場合があります。

1. Amazon MQ コンソールで [Connections] (接続) セクションに移動し、ブローカー `MyBroker1` の ActiveMQ ウェブコンソールエンドポイントをメモします。
2. ブローカー `MyBroker1` の ActiveMQ ウェブコンソールに移動します。
3. ネットワークブリッジが接続されていることを確認するには、[ネットワーク] を選択します。

   [ネットワークブリッジ] セクションで、`MyBroker2` の名前とアドレスが [リモートブローカー] と [リモートアドレス] の列にリストされます。
4. ブローカー `MyBroker2` にアクセスできる任意のマシンから、コンシューマーを作成します。以下はその例です。

   ```
   activemq consumer --brokerUrl "ssl://
   b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-east-2.amazonaws.com:61617" \
    --user commonUser \
    --password myPassword456 \
    --destination queue://MyQueue
   ```

   コンシューマーは `MyBroker2` の OpenWire エンドポイントに接続し、キュー `MyQueue` からメッセージを消費し始めます。
5. ブローカー `MyBroker1` にアクセスできる任意のマシンから、プロデューサーを作成し、いくつかのメッセージを送信します。以下はその例です。

```
activemq producer --brokerUrl "ssl://
b-9876l5k4-32ji-109h-8gfe-7d65c4b132a1-1.mq.us-east-2.amazonaws.com:61617" \
 --user commonUser \
 --password myPassword456 \
 --destination queue://MyQueue \
 --persistent true \
 --messageSize 1000 \
 --messageCount 10000
```

プロデューサーは MyBroker1 の OpenWire エンドポイントに接続し、キュー MyQueue に永続的なメッセージを生成し始めます。

Amazon MQ ブローカーへの Java アプリケーションの接続

Amazon MQ ActiveMQ ブローカーを作成したら、ブローカーにアプリケーションを接続できます。以下の例では、Java Message Service (JMS) を使用してブローカーへの接続を作成し、キューを作成して、メッセージを送信する方法を説明します。完全な Java の実用例については、「Working Java Example」を参照してください。

ActiveMQ ブローカーには、さまざまな ActiveMQ クライアントを使用して接続できます。ActiveMQ クライアントを使用することをお勧めします。

トピック

- 前提条件
- メッセージプロデューサーを作成してメッセージを送信する
- メッセージコンシューマーを作成してメッセージを受信する

前提条件

VPC 属性 を有効にする

VPC 内でブローカーにアクセスできることを確実にするには、enableDnsHostnames および enableDnsSupport VPC 属性を有効にする必要があります。詳細については、Amazon VPC ユーザーガイドの「VPC の DNS サポート」を参照してください。

インバウンド接続を有効にする

次に、アプリケーションのインバウンド接続を有効にします。

1. Amazon MQ コンソールにサインインします。
2. ブローカーのリストからブローカーの名前 (MyBroker など) を選択します。
3. [***MyBroker***] ページの [Connections] (接続) セクションで、ブローカーのウェブコンソール URL とワイヤレベルプロトコルのアドレスとポートをメモします。
4. [Details] (詳細) セクションの [Security and network] (セキュリティとネットワーク) で、セキュリティグループの名前または
   
   をクリックします。

   EC2 ダッシュボードの [セキュリティグループ] ページが表示されます。
5. セキュリティグループのリストから、セキュリティグループを選択します。
6. ページ下部で、[インバウンド] を選択し、次に [編集] を選択します。
7. [Edit inbound rules] (インバウンドルールの編集) ダイアログボックスで、パブリックアクセスを許可する URL またはエンドポイントごとにルールを追加します (以下の例は、これをブローカーのウェブコンソールに対して行う方法を説明しています)。

   a. [ルールの追加] を選択します。
   b. [タイプ] で、[カスタム TCP] を選択します。
   c. [Port Range] (ポート範囲) にはウェブコンソールポート (8162) を入力します。
   d. [Source] (ソース) では、[Custom] (カスタム) が選択された状態のままにしておき、ウェブコンソールにアクセスできるようにするシステムの IP アドレスを入力します (`192.0.2.1` など)。
   e. [Save] (保存) をクリックします。

      これで、ブローカーはインバウンド接続を受け入れることができます。

Java の依存関係を追加する

`activemq-client.jar` パッケージと `activemq-pool.jar` パッケージを Java クラスパスに追加します。以下の例は、Maven プロジェクトの `pom.xml` ファイルにあるこれらの依存関係を示しています。

```
<dependencies>
    <dependency>
        <groupId>org.apache.activemq</groupId>
        <artifactId>activemq-client</artifactId>
```

```
        <version>5.15.16</version>
    </dependency>
    <dependency>
        <groupId>org.apache.activemq</groupId>
        <artifactId>activemq-pool</artifactId>
        <version>5.15.16</version>
    </dependency>
</dependencies>
```

`activemq-client.jar` の詳細については、Apache ActiveMQ ドキュメントの「Initial Configuration」を参照してください。

> **Important**
>
> 以下のコード例では、プロデューサーとコンシューマーが単一のスレッド内で実行されます。実稼働システム (またはブローカーインスタンスのフェイルオーバーをテストする) には、プロデューサーとコンシューマーが個別のホストまたはスレッドで実行されるようにしてください。

メッセージプロデューサーを作成してメッセージを送信する

メッセージプロデューサーを作成してメッセージを受信するには、次の手順に従います。

1. ブローカーのエンドポイントを使用してメッセージプロデューサーの JMS プール接続ファクトリを作成してから、ファクトリに対して `createConnection` メソッドを呼び出します。

   > **Note**
   >
   > アクティブ/スタンバイブローカーの場合、Amazon MQ は 2 つの ActiveMQ ウェブコンソール URL を提供しますが、一度に 1 つの URL しかアクティブになりません。同様に、Amazon MQ はワイヤレベルプロトコルごとに 2 つのエンドポイントを提供しますが、ペアごとに一度に 1 つのエンドポイントしかアクティブになりません。`-1` および `-2` サフィックスは冗長ペアを表します。詳細については、「Amazon MQ for ActiveMQ ブローカーのデプロイオプション」を参照してください。
   > ワイヤレベルプロトコルのエンドポイントについては、フェイルオーバートランスポートを使用することによって、アプリケーションがエンドポイントのどちらか一方に接続することを許可する必要があります。

```
// Create a connection factory.
final ActiveMQConnectionFactory connectionFactory = new
 ActiveMQConnectionFactory(wireLevelEndpoint);

// Pass the sign-in credentials.
connectionFactory.setUserName(activeMqUsername);
connectionFactory.setPassword(activeMqPassword);

// Create a pooled connection factory.
final PooledConnectionFactory pooledConnectionFactory = new
 PooledConnectionFactory();
pooledConnectionFactory.setConnectionFactory(connectionFactory);
pooledConnectionFactory.setMaxConnections(10);

// Establish a connection for the producer.
final Connection producerConnection = pooledConnectionFactory.createConnection();
producerConnection.start();

// Close all connections in the pool.
pooledConnectionFactory.clear();
```

> **Note**
>
> メッセージプロデューサーは、常に PooledConnectionFactory クラスを使用する必要があります。詳細については、「常に接続プールを使用する」を参照してください。

2. セッション、MyQueue という名前のキュー、およびメッセージプロデューサーを作成します。

```
// Create a session.
final Session producerSession = producerConnection.createSession(false,
 Session.AUTO_ACKNOWLEDGE);

// Create a queue named "MyQueue".
final Destination producerDestination = producerSession.createQueue("MyQueue");

// Create a producer from the session to the queue.
final MessageProducer producer =
 producerSession.createProducer(producerDestination);
producer.setDeliveryMode(DeliveryMode.NON_PERSISTENT);
```

3. メッセージ文字列 "Hello from Amazon MQ!" を作成してから、メッセージを送信します。

```
// Create a message.
final String text = "Hello from Amazon MQ!";
TextMessage producerMessage = producerSession.createTextMessage(text);

// Send the message.
producer.send(producerMessage);
System.out.println("Message sent.");
```

4. プロデューサーをクリーンアップします。

```
producer.close();
producerSession.close();
producerConnection.close();
```

メッセージコンシューマーを作成してメッセージを受信する

メッセージプロデューサーを作成してメッセージを受信するには、次の手順に従います。

1. ブローカーのエンドポイントを使用してメッセージプロデューサーの JMS 接続ファクトリを作成してから、ファクトリに対して createConnection メソッドを呼び出します。

```
// Create a connection factory.
final ActiveMQConnectionFactory connectionFactory = new
 ActiveMQConnectionFactory(wireLevelEndpoint);

// Pass the sign-in credentials.
connectionFactory.setUserName(activeMqUsername);
connectionFactory.setPassword(activeMqPassword);

// Establish a connection for the consumer.
final Connection consumerConnection = connectionFactory.createConnection();
consumerConnection.start();
```

> **Note**
>
> メッセージコンシューマーには、PooledConnectionFactory クラスを一切使用しないでください。詳細については、「常に接続プールを使用する」を参照してください。

2. セッション、MyQueue という名前のキュー、およびメッセージコンシューマーを作成します。

```
// Create a session.
final Session consumerSession = consumerConnection.createSession(false,
 Session.AUTO_ACKNOWLEDGE);

// Create a queue named "MyQueue".
final Destination consumerDestination = consumerSession.createQueue("MyQueue");

// Create a message consumer from the session to the queue.
final MessageConsumer consumer =
 consumerSession.createConsumer(consumerDestination);
```

3. メッセージの待機を開始し、メッセージの到着時にメッセージを受信します。

```
// Begin to wait for messages.
final Message consumerMessage = consumer.receive(1000);

// Receive the message when it arrives.
final TextMessage consumerTextMessage = (TextMessage) consumerMessage;
System.out.println("Message received: " + consumerTextMessage.getText());
```

> **Note**
>
> AWS メッセージングサービス (Amazon SQS など) とは異なり、コンシューマーは常にブローカーに接続されます。

4. コンシューマー、セッション、および接続を閉じます。

```
consumer.close();
consumerSession.close();
consumerConnection.close();
```

ActiveMQ ブローカーの LDAP との統合

> **Important**
>
> Amazon MQ では、プライベート CA によって発行されたサーバー証明書はサポートされません。

ActiveMQ ブローカーには、TLS が有効化されている以下のプロトコルを使用してアクセスできます。

- AMQP
- MQTT
- MQTT over WebSocket
- OpenWire
- STOMP
- STOMP over WebSocket

Amazon MQ では、ユーザー許可の管理に、ネイティブ ActiveMQ 認証か LDAP 認証と認可のどちらかを選択できます。ActiveMQ のユーザー名とパスワードに関する制限の詳細については、「Users」を参照してください。

ActiveMQ のユーザーおよびグループによるキューとトピックの使用を認可するには、ブローカーの設定を編集する必要があります。Amazon MQ は、ActiveMQ の Simple Authentication Plugin を使用して、送信先に対する読み込みと書き込みを制限します。詳細情報と例については、「認可マップを常に設定する」および「`authorizationEntry`」を参照してください。

> **Note**
>
> 現在、Amazon MQ はクライアント証明書認証をサポートしていません。

トピック

- LDAP を ActiveMQ に統合する
- 前提条件
- LDAP の使用開始
- LDAP 統合の仕組み

LDAP を ActiveMQ に統合する

Amazon MQ ユーザーは、Lightweight Directory Access Protocol (LDAP) サーバーに保存されている認証情報を使用して認証することができます。これを使用して、Amazon MQ ユーザーの追加、削

除、変更、およびトピックとキューへの許可の割り当てを行うことも可能です。ブローカーの作成、更新、および削除といった管理操作には引き続き IAM 認証情報が必要となり、これらは LDAP と統合されません。

LDAP サーバーを使用した Amazon MQ ブローカーの認証と認可の簡素化と一元化を希望するお客様は、この機能を使用できます。すべてのユーザー認証情報を LDAP サーバーに保存することにより、これらの認証情報を保存して管理する一元的な場所が提供されるため、時間と労力を節約できます。

Amazon MQ は、Apache ActiveMQ JAAS プラグインを使用して LDAP サポートを提供します。このプラグインがサポートする LDAP サーバー (Microsoft Active Directory や OpenLDAP など) ならば、Amazon MQ でもサポートされます。プラグインの詳細については、ActiveMQ ドキュメントの「Security」セクションを参照してください。

ユーザーに加えて、特定のグループまたはユーザーのトピックとキューへのアクセスも、LDAP サーバー経由で指定できます。これは、LDAP サーバーでトピックとキューを表すエントリを作成してから、特定の LDAP ユーザーまたはグループに許可を割り当てることで実行します。その後、LDAP サーバーから認可データを取得するようにブローカーを設定できます。

> **Important**
>
> LDAP を使用する場合、認証では大文字と小文字は区別されませんが、認可ではユーザー名の大文字と小文字が区別されます。

前提条件

新規または既存の Amazon MQ ブローカーに LDAP サポートを追加する前に、サービスアカウントをセットアップする必要があります。このサービスアカウントは、LDAP サーバーへの接続を開始するために必要で、この接続を行うために適切な許可を持っている必要があります。このサービスアカウントは、ブローカーの LDAP 認証をセットアップします。後続のクライアント接続は、いずれも同じ接続を介して認証されます。

サービスアカウントは、接続を開始するためのアクセス権を持つ LDAP サーバー内のアカウントです。これは標準の LDAP 要件であり、サービスアカウントの認証情報を提供する必要があるのは 1 度だけです。接続がセットアップされると、その後のすべてのクライアント接続が LDAP サーバー経由で認証されます。サービスアカウントの認証情報は暗号化された形態でセキュアに保存され、Amazon MQ 以外はアクセスできません。

ActiveMQ との統合には、LDAP サーバーに特定のディレクトリ情報ツリー (DIT) が必要です。この構造を明確に示すサンプル `ldif` ファイルについては、ActiveMQ ドキュメントの「Security」セクションで「Import the following LDIF file into the LDAP server」を参照してください。

LDAP の使用開始

使用を開始するには、Amazon MQ コンソールに移動し、新しい Amazon MQ ブローカーインスタンスの作成時または既存のブローカーインスタンスの編集時に [LDAP 認証と認可] を選択します。

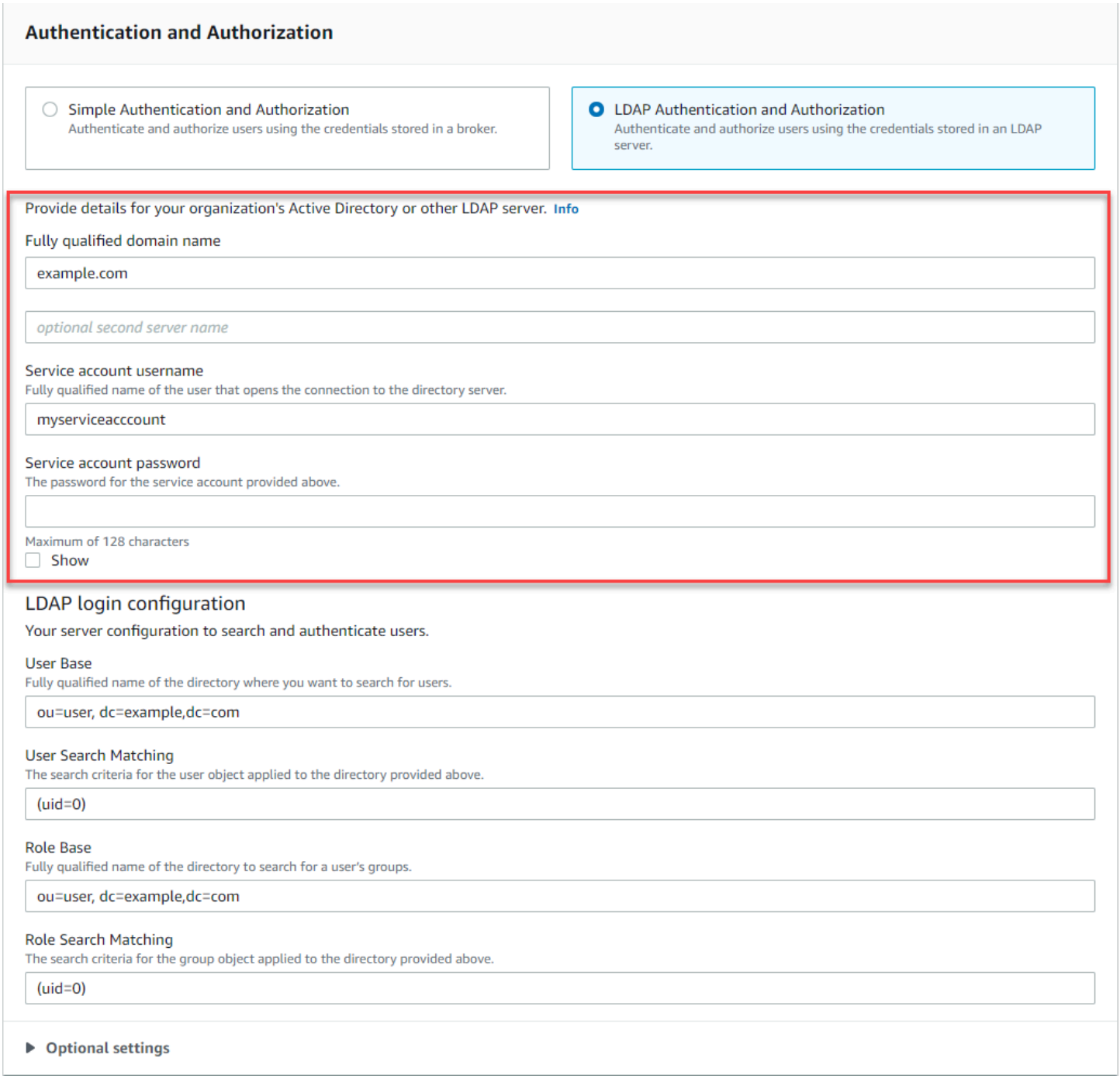
サービスアカウントに関する以下の情報を入力します。

- [完全修飾ドメイン名] 認証リクエストと認可リクエストを発行する先の LDAP サーバーの場所です。

> **Note**
>
> 入力する LDAP サーバーの完全修飾ドメイン名には、プロトコルまたはポート番号を含めないでください。Amazon MQ は、完全修飾ドメイン名の先頭にプロトコル `ldaps` を付加し、末尾にポート番号 636 を付加します。
> 例えば、`example.com` という完全修飾ドメインを指定する場合、Amazon MQ は URL `ldaps://example.com:636` を使用して LDAP サーバーにアクセスします。
> ブローカーホストが LDAP サーバーと正常に通信できるようにするには、完全修飾ドメイン名がパブリックに解決可能である必要があります。LDAP サーバーをプライベートかつセキュアに保つには、サーバーのインバウンドルールでインバウンドトラフィックを制限して、ブローカーの VPC 内からのトラフィックのみを許可します。

- Service account username (サービスアカウントのユーザーネーム) LDAP サーバーへの初期バインドを実行するために使用されるユーザーの識別名です。
- Service account password (サービスアカウントのパスワード) 初期バインドを実行するユーザーのパスワードです。

以下の画像では、これらの詳細情報を指定する場所が強調されています。



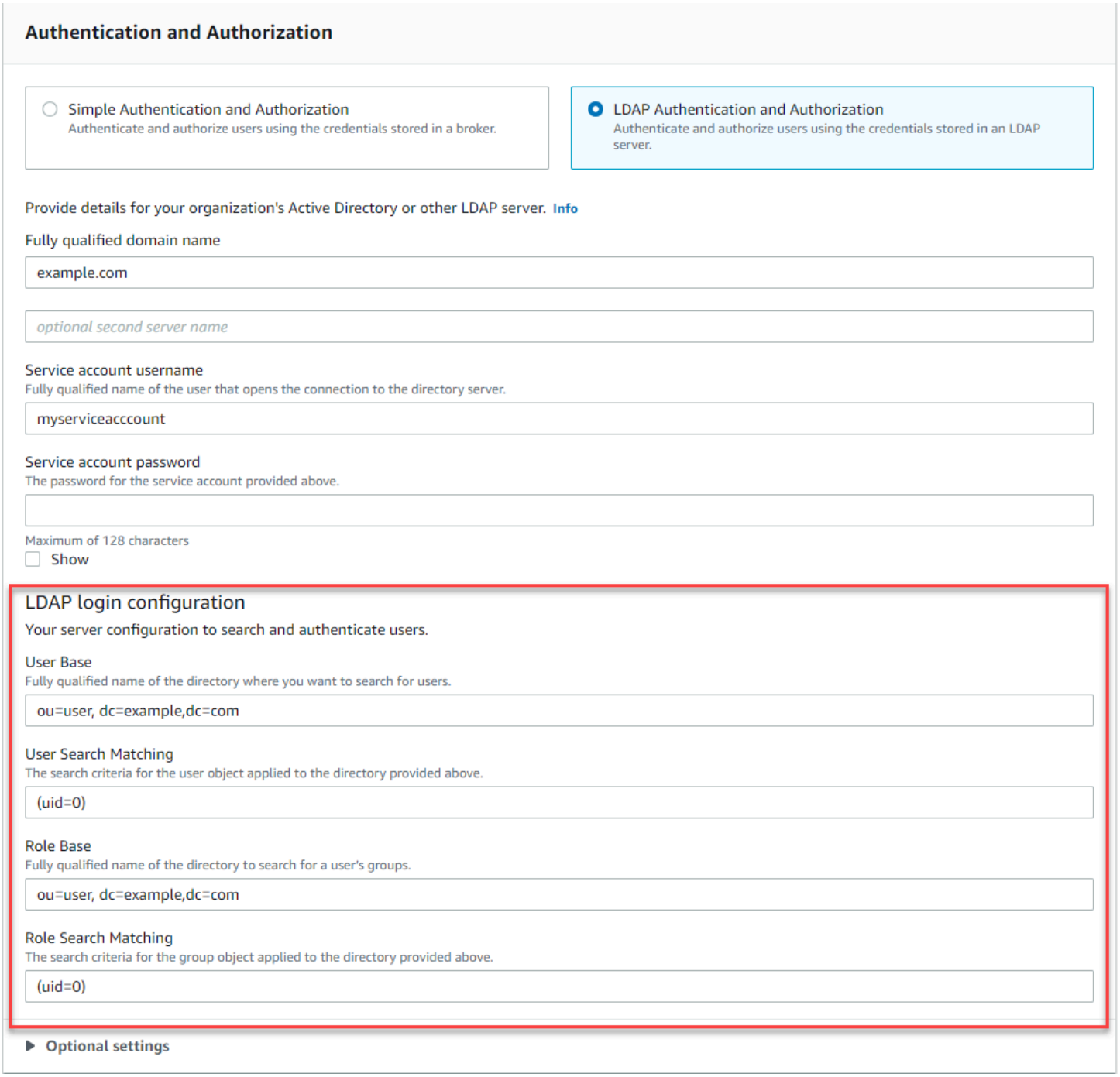
[LDAP login configuration] (LDAP ログイン設定) セクションで、以下の必須情報を入力します。

- User Base (ユーザーベース) ユーザーの検索先となる、ディレクトリ情報ツリー (DIT) 内のノードの識別名です。
- User Search Matching (ユーザー検索のマッチング) `userBase` 内のユーザーを検索するために使用される LDAP 検索フィルターです。検索フィルターの `{0}` プレースホルダーにはクライアント

のユーザーネームが代入されます。詳細については、「認証」および「Authorization」を参照してください。

- Role Base (ロールベース) ロールの検索先となる、DIT 内のノードの識別名です。ロールは、ディレクトリ内の明示的な LDAP グループエントリとして設定できます。一般的なロールエントリは、ロール名の 1 つの属性 (共通名 (CN)など)、もう一つの属性 (`member` など)、およびロールグループに属するユーザーの識別名またはユーザーネームを表す値で構成することができます。例えば、組織単位 `group` がある場合には、識別名 `ou=group,dc=example,dc=com` を指定できます。
- Role Search Matching (ロール検索のマッチング) `roleBase` 内のロールを検索するために使用される LDAP 検索フィルターです。検索フィルターの `{0}` プレースホルダーには、`userSearchMatching` に一致するユーザーの識別名が代入されます。`{1}` プレースホルダーには、クライアントのユーザーネームが代入されます。例えば、ディレクトリ内のロールエントリに `member` という名前の属性が含まれ、そのロール内のすべてのユーザーのユーザーネームが含められている場合は、検索フィルター (`member:=uid={1}`) を指定できます。

以下の画像では、これらの詳細情報を指定する場所が強調されています。



[Optional settings] (オプション設定) セクションでは、以下のオプション情報を指定できます。

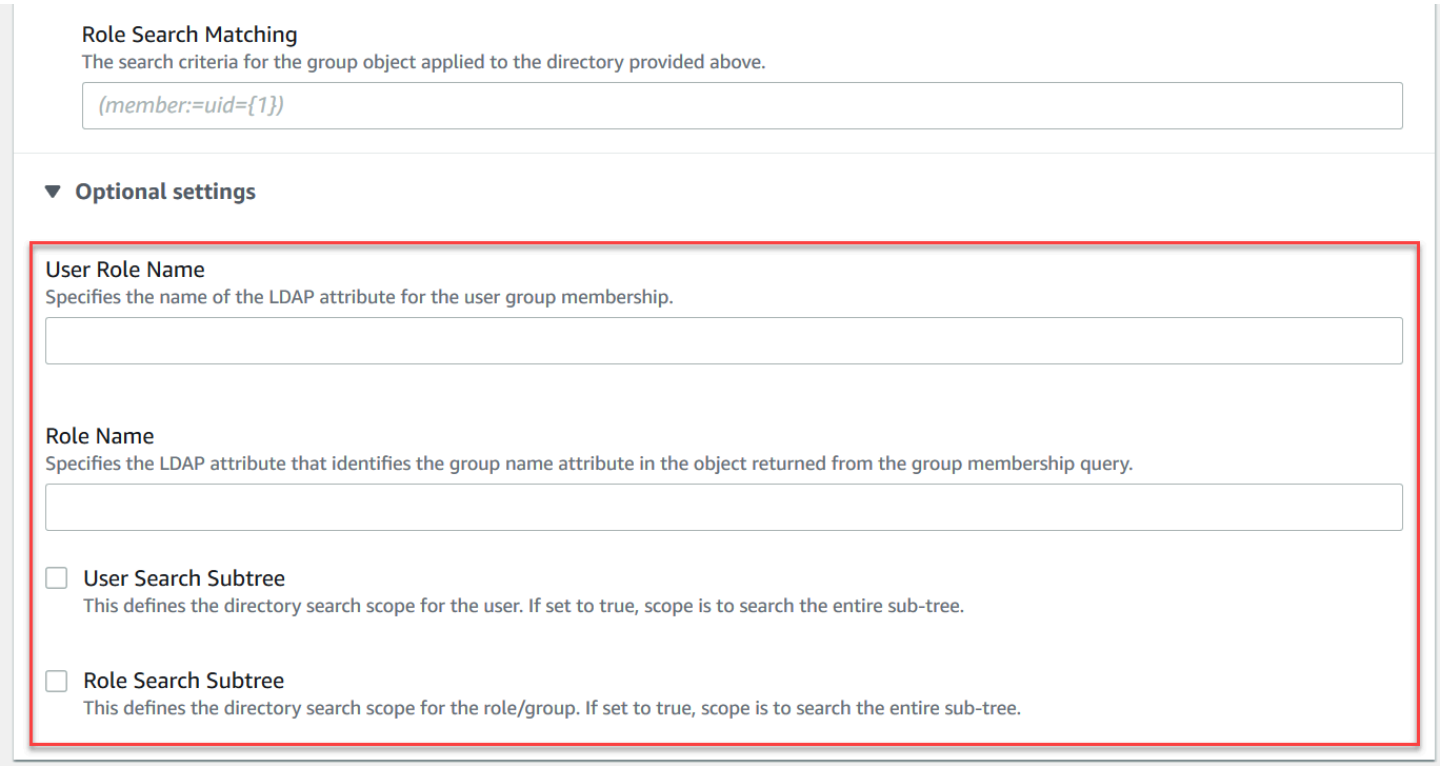
- User Role Name (ユーザーロール名) ユーザーのグループメンバーシップに関するユーザーのディレクトリエントリ内の LDAP 属性の名前です。場合によっては、ユーザーのディレクトリエントリ内の属性の値によって、ユーザーロールを識別できることもあります。`userRoleName` オプションは、この属性の名前を指定することを可能にします。例えば、以下のユーザーエントリについて考えてみましょう。

```
dn: uid=jdoe,ou=user,dc=example,dc=com
objectClass: user
uid: jdoe
sn: jane
cn: Jane Doe
mail: j.doe@somecompany.com
memberOf: role1
userPassword: password
```

上記の例に正しい userRoleName を提供するには、memberOf 属性を指定します。認証が成功すると、ユーザーにロール role1 が割り当てられます。

- Role Name (ロール名) ロールエントリ内のグループ名属性で、値がそのロールの名前になってます。例えば、グループエントリの共通名には cn を指定できます。認証が成功すると、ユーザーには、メンバーになっている各ロールエントリの cn 属性の値が割り当てられます。
- User Search Subtree (ユーザー検索サブツリー) LDAP ユーザー検索クエリの範囲を定義します。true の場合、userBase によって定義されたノード下にあるサブツリー全体を検索するように範囲が設定されます。
- Role Search Subtree (ロール検索サブツリー) LDAP ロール検索クエリの範囲を定義します。true の場合、roleBase によって定義されたノード下にあるサブツリー全体を検索するように範囲が設定されます。

以下の画像では、これらのオプション設定を指定する場所が強調されています。



LDAP 統合の仕組み

統合は、認証の構造と認可の構造という 2 つの主要カテゴリに分けて考えることができます。

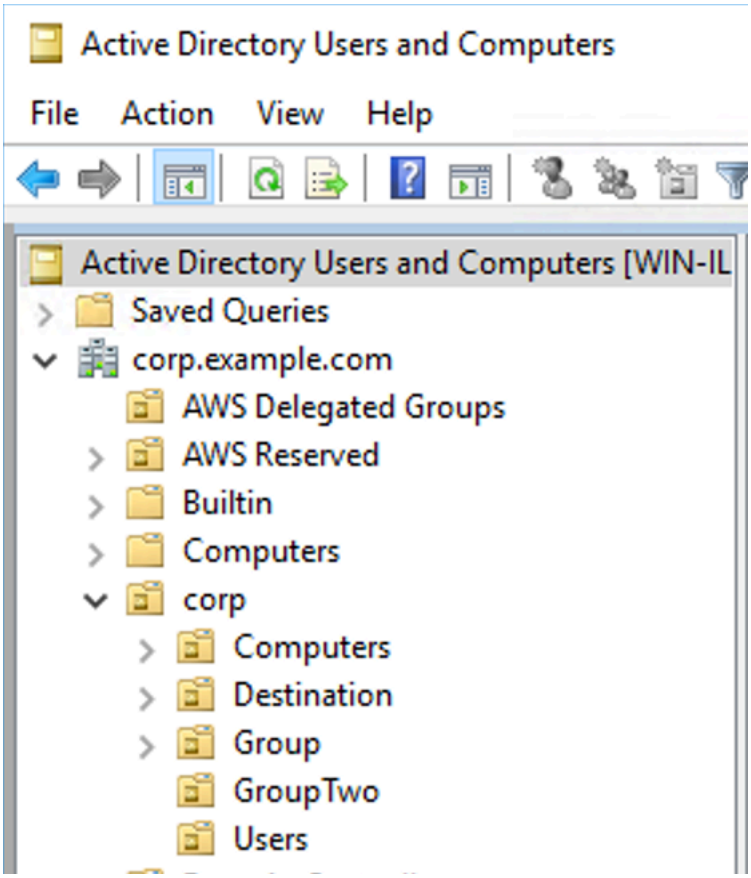
認証

認証では、クライアントの認証情報が有効である必要があります。これらの認証情報は、LDAP サーバーのユーザベース内のユーザーに対して検証されます。

ActiveMQ ブローカーに提供されるユーザーベースは、LDAP サーバーでユーザーが保存されている DIT 内のノードをポイントしている必要があります。たとえば、 を使用していて AWS Managed Microsoft AD、ドメインコンポーネント `corp`、`example`、および `com`、組織単位 `corp` と があるコンポーネント内には`Users`、ユーザーベースとして以下を使用します。

```
OU=Users,OU=corp,DC=corp,DC=example,DC=com
```

ActiveMQ ブローカーは、ブローカーに対するクライアント接続リクエストを認証するために、DIT 内のこの場所でユーザーを検索します。



ActiveMQ ソースコードは、ユーザーの属性名を `uid` にハードコードするため、各ユーザーにこの属性セットがあることを確認する必要があります。簡略化のため、ユーザーの接続ユーザーネームを使用できます。詳細については、activemq ソースコードと「Configuring ID mappings in Active Directory Users and Computers for Windows Server 2016 (and subsequent) versions」を参照してください。

特定のユーザーに対して ActiveMQ コンソールアクセスを有効にするには、ユーザーが `amazonmq-console-admins` グループに属していることを確認してください。

Authorization

認可のため、ブローカーの設定に許可の検索ベースが指定されています。認可は、ブローカーの `activemq.xml` 設定ファイルにある `cachedLdapAuthorizationMap` 要素を通じて、送信先ごと (またはワイルドカード、送信先セット) に行われます。詳細については、「Cached LDAP Authorization Module」を参照してください。

> **Note**
>
> ブローカーactivemq.xmlの設定ファイルで cachedLDAPAuthorizationMap要素を使用するには、 を使用して設定を作成する AWS マネジメントコンソールときに LDAP 認証および認可オプションを選択するか、 を使用して設定を作成する AWS マネジメントコンソールときに を設定するか、Amazon MQ API を使用して新しい設定を作成するLDAPときに authenticationStrategyプロパティを に設定する必要があります。

cachedLDAPAuthorizationMap 要素の一部として、以下の 3 つの属性を指定する必要があります。

- queueSearchBase
- topicSearchBase
- tempSearchBase

> **Important**
>
> ブローカーの設定ファイルに機密情報が直接配置されることを防ぐため、Amazon MQ は cachedLdapAuthorizationMap での以下の属性の使用をブロックします。
>
> - connectionURL
> - connectionUsername
> - connectionPassword
>
> ブローカーを作成すると、Amazon MQ は、 を介して AWS マネジメントコンソール、または API リクエストの ldapServerMetadataプロパティで指定した値を上記の属性に置き換えます。

以下は、cachedLdapAuthorizationMap の実用例です。

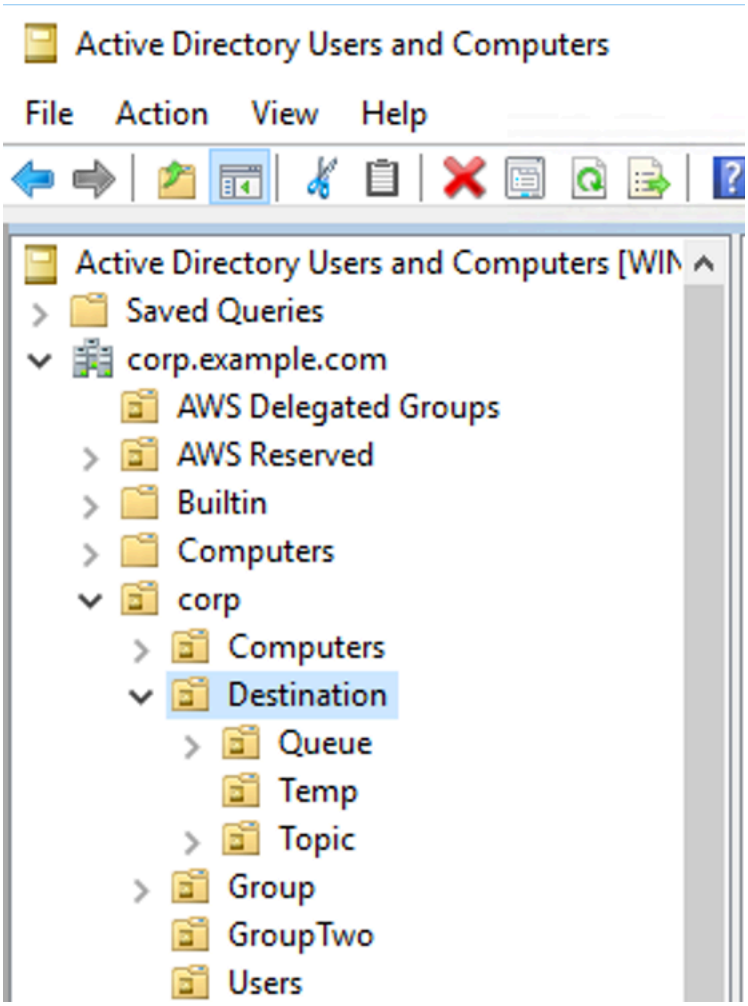
```
<authorizationPlugin>
    <map>
        <cachedLDAPAuthorizationMap
            queueSearchBase="ou=Queue,ou=Destination,ou=corp,dc=corp,dc=example,dc=com"
            topicSearchBase="ou=Topic,ou=Destination,ou=corp,dc=corp,dc=example,dc=com"
```

```
            tempSearchBase="ou=Temp,ou=Destination,ou=corp,dc=corp,dc=example,dc=com"
            refreshInterval="300000"
            legacyGroupMapping="false"
        />
    </map>
</authorizationPlugin>
```

これらの値は、送信先の各タイプに対する許可が指定されている、DIT 内の場所を特定します。したがって、上記の例では AWS Managed Microsoft AD、、corp、exampleおよび の同じドメインコンポーネントを使用してcom、すべての送信先タイプを含むdestinationように という名前の組織単位を指定します。その OU 内で、queues、topics、および temp の各送信先の OU を作成します。

これは、Queue タイプの送信先の認可情報を提供するキュー検索ベースの場所が、DIT 内の以下の場所になることを意味します。

```
OU=Queue,OU=Destination,OU=corp,DC=corp,DC=example,DC=com
```



同様に、Topics および Temp 送信先の許可ルールの場所も、DIT 内の同じレベルになります。

```
OU=Topic,OU=Destination,OU=corp,DC=corp,DC=example,DC=com
OU=Temp,OU=Destination,OU=corp,DC=corp,DC=example,DC=com
```

各送信先タイプ (Queue、Topic、Temp) の OU 内には、ワイルドカードまたは特定の送信先名を指定できます。例えば、プレフィックス DEMO.EVENTS.$. で始まるすべてのキューの認可ルールを提供するには、以下の OU を作成できます。

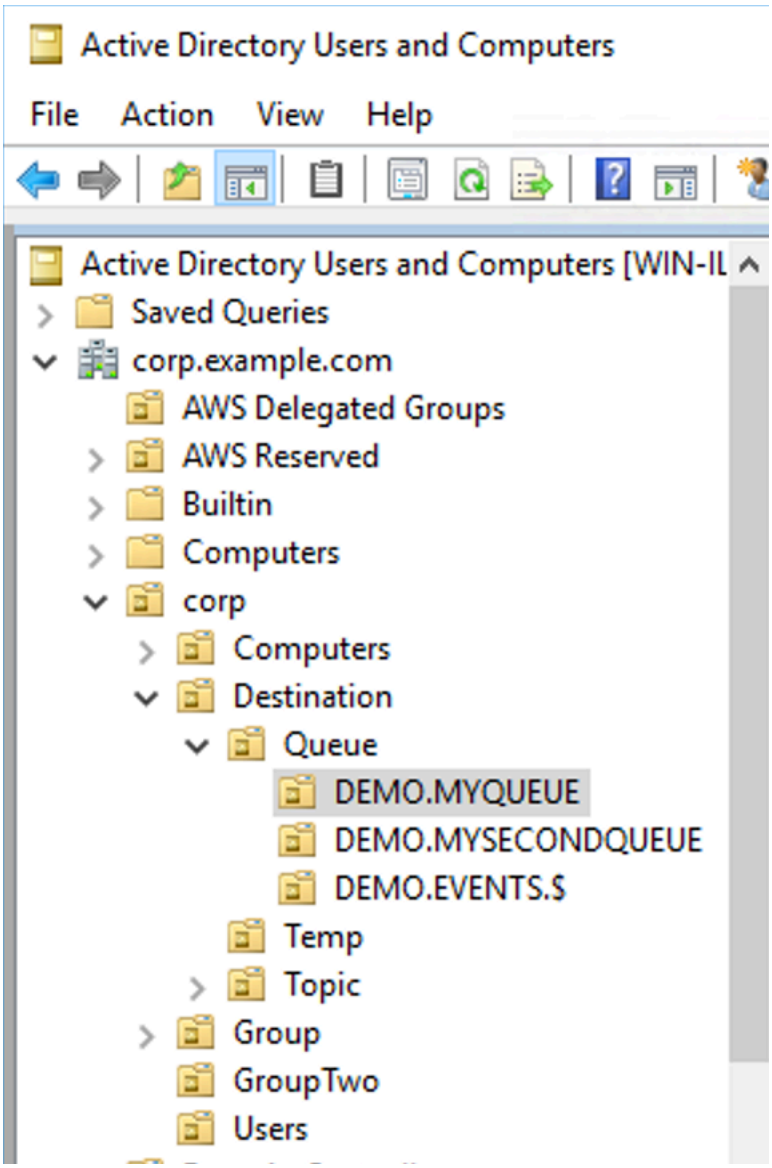
```
OU=DEMO.EVENTS.$,OU=Queue,OU=Destination,OU=corp,DC=corp,DC=example,DC=com
```

> **Note**
>
> DEMO.EVENTS.$ OU は Queue OU 内にあります。

ActiveMQ でのワイルドカードの詳細については、「Wildcards」を参照してください。

DEMO.MYQUEUE などの特定のキューの認可ルールを提供するには、以下のように指定します。

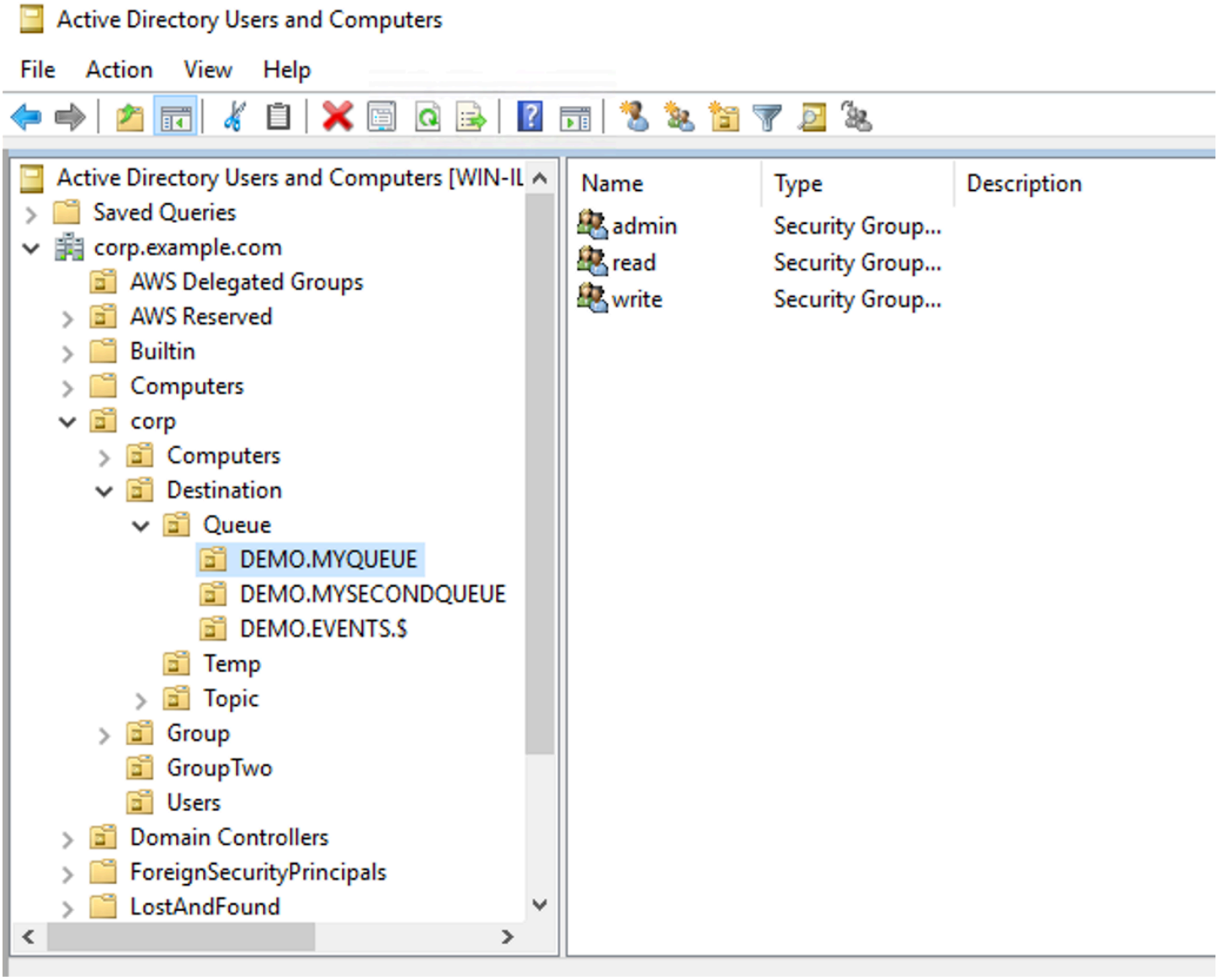
```
OU=DEMO.MYQUEUE,OU=Queue,OU=Destination,OU=corp,DC=corp,DC=example,DC=com
```



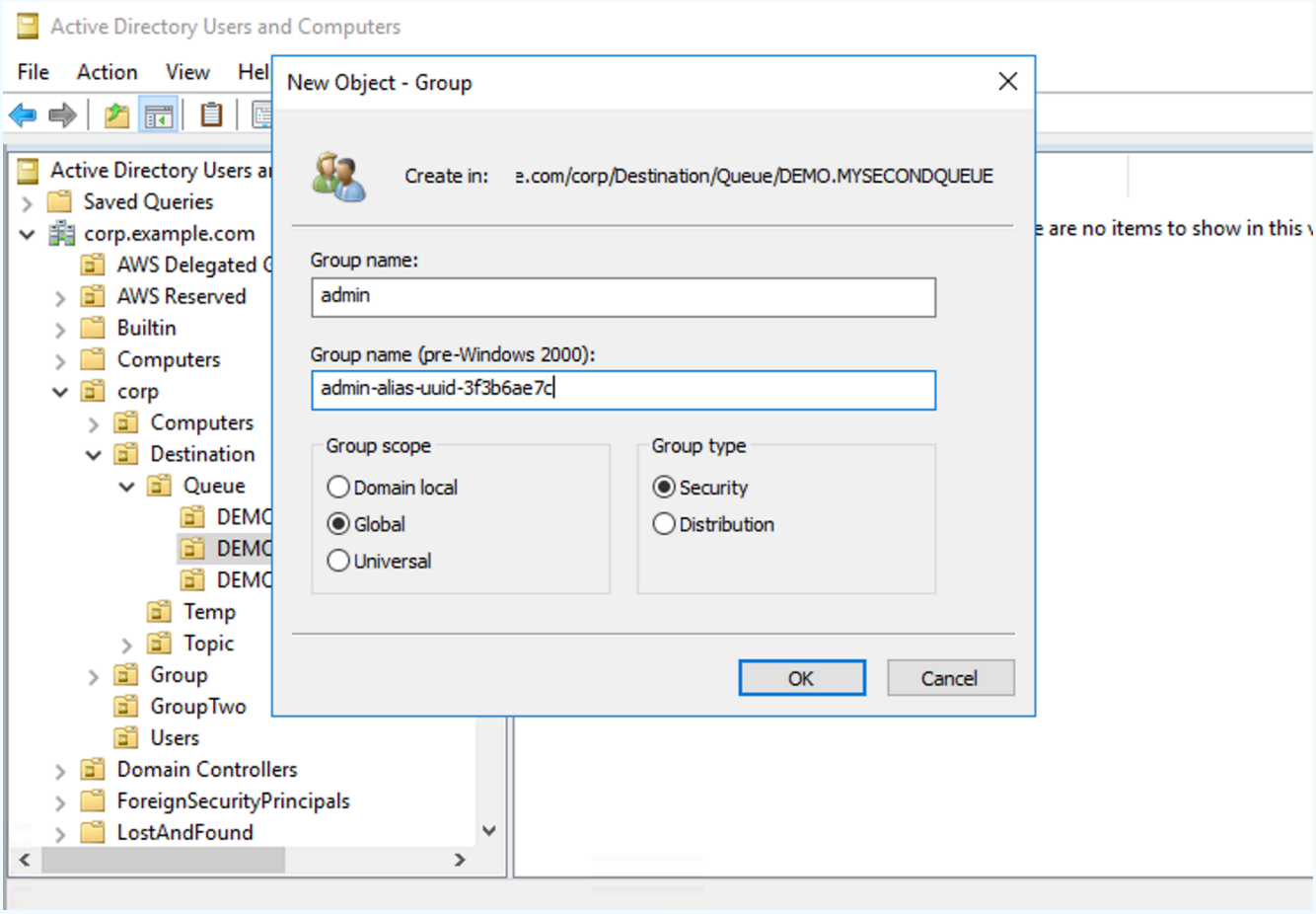
セキュリティグループ

送信先またはワイルドカードを表す各 OU 内には、3 つのセキュリティグループを作成する必要があります。ActiveMQ のすべての許可と同様に、これらは読み取り/書き込み/管理者許可です。これらの許可のそれぞれがユーザーに許可する操作の詳細については、ActiveMQ ドキュメントの「Security」を参照してください。

これらのセキュリティグループには、`read`、`write`、および `admin` という名前を付ける必要があります。これらの各セキュリティグループ内でユーザーまたはグループを追加することができ、そうすることで、そのユーザーとグループが関連付けられたアクションを実行する許可を得ます。これらのセキュリティグループは、各ワイルドカード送信先セット、または個々の送信先に必要になります。



> **Note**
>
> 管理グループを作成すると、グループ名で競合が発生します。この競合は、Windows 2000 より前のレガシールールが、グループによる同一名の共有を、グループが DIT 内の別の場所にある場合でも許可しないために発生します。[Windows 2000 より前] テキストボックス内の値はセットアップに影響しませんが、グローバルに一意である必要があります。この競合を回避するには、各 admin グループに uuid サフィックスを追加できます。



特定の送信先の admin セキュリティグループにユーザーを追加すると、ユーザーがそのトピックの作成および削除を実行できるようになります。ユーザーを read セキュリティグループに追加すると、送信先からの読み取りが可能になり、write グループに追加すると、送信先への書き込みが可能になります。

セキュリティグループ許可に個々のユーザーを追加することに加えて、グループ全体を追加することもできますが、ActiveMQ はグループの属性名をハードコードするため、activemq ソースコードにあ

るように、追加するグループにオブジェクトクラス `groupOfNames` があることを確実にする必要があります。

これを行うには、ユーザーの `uid` と同じプロセスに従ってください。「Configuring ID mappings in Active Directory Users and Computers for Windows Server 2016 (and subsequent) versions」を参照してください。

ステップ 3: (オプション) AWS Lambda 関数に接続する

AWS Lambda は Amazon MQ ブローカーに接続して、Amazon MQ ブローカーからのメッセージを消費できます。ブローカーを Lambda に接続するときは、キューからメッセージを読み取り、関数 synchronously を呼び出すイベントソースマッピングを作成します。作成するイベントソースマッピングは、ブローカーからメッセージをバッチで読み取り、それらを JSON オブジェクト形式の Lambda ペイロードに変換します。

ブローカーを Lambda 関数に接続する

1. Lambda 関数 execution role に以下の IAM ロール許可を追加します。

   - mq:DescribeBroker
   - ec2:CreateNetworkInterface
   - ec2:DeleteNetworkInterface
   - ec2:DescribeNetworkInterfaces
   - ec2:DescribeSecurityGroups
   - ec2:DescribeSubnets
   - ec2:DescribeVpcs
   - logs:CreateLogGroup
   - logs:CreateLogStream
   - logs:PutLogEvents
   - secretsmanager:GetSecretValue

   > **Note**
   >
   > 必要な IAM 許可がない場合、関数は Amazon MQ リソースからレコードを正常に読み取ることができません。

2. (オプション) パブリックアクセシビリティがないブローカーを作成した場合は、次のいずれかを実行して、Lambda のブローカーへの接続を許可する必要があります。

   - パブリックサブネットごとに 1 つの NAT ゲートウェイを設定します。詳細については、AWS Lambda デベロッパーガイドの「VPC に接続した関数のインターネットアクセスとサービスアクセス」を参照してください。
   - VPC エンドポイントを使用して、Amazon Virtual Private Cloud (Amazon VPC) と Lambda 間の接続を作成します。Amazon VPC は、 AWS Security Token Service (AWS STS) および Secrets Manager エンドポイントにも接続する必要があります。詳細については、AWS Lambda デベロッパーガイドの「Lambda のインターフェイス VPC エンドポイントの設定」を参照してください。

3. AWS マネジメントコンソールを使用して、Lambda 関数のイベントソースとしてブローカーを設定します。`create-event-source-mapping` AWS Command Line Interface コマンドを使用することもできます。

4. ブローカーから取り込まれたメッセージを処理するための Lambda 関数のコードをいくつか記述します。イベントソースマッピングによって取得される Lambda ペイロードは、ブローカーのエンジンタイプに依存します。以下は、Amazon MQ for ActiveMQ キューの Lambda ペイロードの例です。

> **Note**
>
> この例では、`testQueue` がキューの名前です。

```
{
  "eventSource": "aws:amq",
  "eventSourceArn": "arn:aws:mq:us-
west-2:112556298976:broker:test:b-9bcfa592-423a-4942-879d-eb284b418fc8",
  "messages": {
    [
      {
        "messageID": "ID:b-9bcfa592-423a-4942-879d-eb284b418fc8-1.mq.us-
west-2.amazonaws.com-37557-1234520418293-4:1:1:1:1",
        "messageType": "jms/text-message",
        "data": "QUJDOkFBQUE=",
        "connectionId": "myJMSCoID",
        "redelivered": false,
        "destination": {
          "physicalname": "testQueue"
```

```
        },
        "timestamp": 1598827811958,
        "brokerInTime": 1598827811958,
        "brokerOutTime": 1598827811959
      },
      {
        "messageID": "ID:b-9bcfa592-423a-4942-879d-eb284b418fc8-1.mq.us-
west-2.amazonaws.com-37557-1234520418293-4:1:1:1:1",
        "messageType":"jms/bytes-message",
        "data": "3DTOOW7crj51prgVLQaGQ82S48k=",
        "connectionId": "myJMSCoID1",
        "persistent": false,
        "destination": {
          "physicalname": "testQueue"
        },
        "timestamp": 1598827811958,
        "brokerInTime": 1598827811958,
        "brokerOutTime": 1598827811959
      }
    ]
  }
}
```

Amazon MQ の Lambda への接続、Amazon MQ イベントソースに対して Lambda がサポートするオプション、およびイベントソースマッピングエラーの詳細については、AWS Lambda デベロッパーガイドの「Amazon MQ で Lambda を使用する」を参照してください。

ActiveMQ ブローカーユーザーの作成

ActiveMQ ユーザーとは、ActiveMQ ブローカーのキューとトピックにアクセスできる人物またはアプリケーションです。ユーザーは、特定の許可を持つように設定できます。例えば、一部のユーザーに ActiveMQ ウェブコンソールへのアクセスを許可することができます。

グループはセマンティックラベルです。グループをユーザーに割り当てて、グループが特定のキューとトピックに対する送信、受信、管理を行うための許可を設定できます。

> **Note**
>
> グループをユーザーと個別に設定することはできません。グループラベルは、グループに少なくとも 1 人のユーザーを追加するときに作成され、そこからすべてのユーザーを削除するとグループも削除されます。

> **Note**
>
> Amazon MQ 上の ActiveMQ の `activemq-webconsole` グループには、すべてのキューとトピックに対する管理者アクセス許可があります。このグループのすべてのユーザーは管理者アクセス権を持ちます。

以下の例では、 AWS マネジメントコンソールを使用して Amazon MQ ブローカーユーザーを作成、編集、および削除する方法を説明します。

新しい ActiveMQ ブローカーユーザーの作成

1. Amazon MQ コンソールにサインインします。
2. ブローカーリストからブローカーの名前 (MyBroker など) を選択して、[View details] (詳細を表示) をクリックします。

   [***MyBroker***] ページの [Users] (ユーザー) セクションに、このブローカーのすべてのユーザーがリストされます。

	Username	Console access	Groups	Pending modifications
○	paolo.santos	No	Devs	
○	jane.doe	Yes	Admins	

3. [ユーザーの作成] を選択します。
4. [ユーザーの作成] ダイアログボックスに、[ユーザー名] と [パスワード] を入力します。
5. (省略可能) ユーザーが属するグループの名前をコンマで区切って入力します (例: `Devs,Admins`)。
6. (省略可能) ユーザーが ActiveMQ ウェブコンソールにアクセスできるようにするには、[ActiveMQ ウェブコンソール] を選択します。

7. [Create user] (ユーザーの作成) をクリックします。

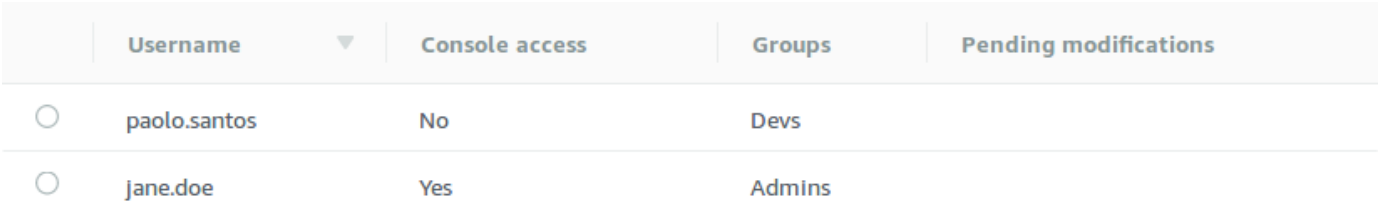
> **Important**
>
> ユーザーを変更しても、その変更はユーザーに直ちに適用されません。変更を適用するには、次のメンテナンスウィンドウまで待機するか、ブローカーを再起動する必要があります。

ActiveMQ ブローカーユーザーの編集

既存のユーザーを使用するには、以下の操作を行います。

1. Amazon MQ コンソールにサインインします。
2. ブローカーリストからブローカーの名前 (MyBroker など) を選択して、[View details] (詳細を表示) をクリックします。

   [***MyBroker***] ページの [Users] (ユーザー) セクションに、このブローカーのすべてのユーザーがリストされます。



3. サインイン認証情報を指定し、[編集] を選択します。

   [ユーザーの編集] ダイアログボックスが表示されます。
4. (省略可能) 新しい [パスワード] を入力します。
5. (省略可能) ユーザーが属するグループの名前をコンマで区切って追加または削除します (例: `Managers, Admins`)。
6. (省略可能) ユーザーが ActiveMQ ウェブコンソールにアクセスできるようにするには、[ActiveMQ ウェブコンソール] を選択します。
7. ユーザーに対する変更を保存するには、[完了] を選択します。

> **Important**
>
> ユーザーを変更しても、その変更はユーザーに直ちに適用されません。変更を適用するには、次のメンテナンスウィンドウまで待機するか、ブローカーを再起動する必要があります。

ActiveMQ ブローカーユーザーの削除

不要になったユーザーは削除できます。

1. Amazon MQ コンソールにサインインします。
2. ブローカーリストからブローカーの名前 (MyBroker など) を選択して、[View details] (詳細を表示) をクリックします。

   [***MyBroker***] ページの [Users] (ユーザー) セクションに、このブローカーのすべてのユーザーがリストされます。

	Username	Console access	Groups	Pending modifications
○	paolo.santos	No	Devs	
○	jane.doe	Yes	Admins	

3. サインイン認証情報 (***MyUser*** など) を指定し、[削除] を選択します。
4. ユーザーの削除を確認するには、[Delete ***MyUser***? (MyUser を削除しますか?)] ダイアログボックスで、[削除] を選択します。

> **Important**
>
> ユーザーを変更しても、その変更はユーザーに直ちに適用されません。変更を適用するには、次のメンテナンスウィンドウまで待機するか、ブローカーを再起動する必要があります。

ActiveMQ での Java Message Service (JMS) の使用の実用例

以下の例で、プログラムで ActiveMQ を操作する方法を示します。

- OpenWire の Java コードの例は、ブローカーに接続し、キューを作成して、メッセージを送受信します。詳細および説明については、「Connecting a Java application to your broker」を参照してください。
- MQTT のサンプル Java コードは、ブローカーへの接続、トピックの作成、およびメッセージの発行と受信を行います。
- STOMP+WSS のサンプル Java コードは、ブローカーへの接続、キューの作成、およびメッセージの発行と受信を行います。

前提条件

VPC 属性 を有効にする

VPC 内でブローカーにアクセスできることを確実にするには、`enableDnsHostnames` および `enableDnsSupport` VPC 属性を有効にする必要があります。詳細については、Amazon VPC ユーザーガイドの「VPC の DNS サポート」を参照してください。

インバウンド接続を有効にする

Amazon MQ をプログラムで操作するには、インバウンド接続を使用する必要があります。

1. Amazon MQ コンソールにサインインします。
2. ブローカーのリストからブローカーの名前 (MyBroker など) を選択します。
3. [***MyBroker***] ページの [Connections] (接続) セクションで、ブローカーのウェブコンソール URL とワイヤレベルプロトコルのアドレスとポートをメモします。
4. [Details] (詳細) セクションの [Security and network] (セキュリティとネットワーク) で、セキュリティグループの名前または
   
   をクリックします。

   EC2 ダッシュボードの [セキュリティグループ] ページが表示されます。
5. セキュリティグループのリストから、セキュリティグループを選択します。
6. ページ下部で、[インバウンド] を選択し、次に [編集] を選択します。
7. [Edit inbound rules] (インバウンドルールの編集) ダイアログボックスで、パブリックアクセスを許可する URL またはエンドポイントごとにルールを追加します (以下の例は、これをブローカーのウェブコンソールに対して行う方法を説明しています)。

   a. [ルールの追加] を選択します。

b. [タイプ] で、[カスタム TCP] を選択します。

c. [Port Range] (ポート範囲) にはウェブコンソールポート (8162) を入力します。

d. [Source] (ソース) では、[Custom] (カスタム) が選択された状態のままにしておき、ウェブコンソールにアクセスできるようにするシステムの IP アドレスを入力します (`192.0.2.1` など)。

e. [Save] (保存) をクリックします。

これで、ブローカーはインバウンド接続を受け入れることができます。

Java の依存関係を追加する

OpenWire

`activemq-client.jar` パッケージと `activemq-pool.jar` パッケージを Java クラスパスに追加します。以下の例は、Maven プロジェクトの `pom.xml` ファイルにあるこれらの依存関係を示しています。

```
<dependencies>
    <dependency>
        <groupId>org.apache.activemq</groupId>
        <artifactId>activemq-client</artifactId>
        <version>5.15.16</version>
    </dependency>
    <dependency>
        <groupId>org.apache.activemq</groupId>
        <artifactId>activemq-pool</artifactId>
        <version>5.15.16</version>
    </dependency>
</dependencies>
```

`activemq-client.jar` の詳細については、Apache ActiveMQ ドキュメントの「Initial Configuration」を参照してください。

MQTT

`org.eclipse.paho.client.mqttv3.jar` パッケージを Java クラスパスに追加します。次の例では、この依存関係を Maven プロジェクトの `pom.xml` ファイルで示しています。

```
<dependencies>
                    <dependency>
```

```
                        <groupId>org.eclipse.paho</groupId>
                        <artifactId>org.eclipse.paho.client.mqttv3</artifactId>
                        <version>1.2.0</version>
                    </dependency>
                    </dependencies>
```

org.eclipse.paho.client.mqttv3.jar の詳細については、Eclipse Paho Java Client を参照してください。

STOMP+WSS

次のパッケージを Java クラスパスに追加しました。

- spring-messaging.jar
- spring-websocket.jar
- javax.websocket-api.jar
- jetty-all.jar
- slf4j-simple.jar
- jackson-databind.jar

以下の例は、Maven プロジェクトの pom.xml ファイルにあるこれらの依存関係を示しています。

```
<dependencies>
                    <dependency>
                        <groupId>org.springframework</groupId>
                        <artifactId>spring-messaging</artifactId>
                        <version>5.0.5.RELEASE</version>
                    </dependency>
                    <dependency>
                        <groupId>org.springframework</groupId>
                        <artifactId>spring-websocket</artifactId>
                        <version>5.0.5.RELEASE</version>
                    </dependency>
                    <dependency>
                        <groupId>javax.websocket</groupId>
                        <artifactId>javax.websocket-api</artifactId>
                        <version>1.1</version>
                    </dependency>
                    <dependency>
                        <groupId>org.eclipse.jetty.aggregate</groupId>
```

```
                    <artifactId>jetty-all</artifactId>
                    <type>pom</type>
                    <version>9.3.3.v20150827</version>
                </dependency>
                <dependency>
                    <groupId>org.slf4j</groupId>
                    <artifactId>slf4j-simple</artifactId>
                    <version>1.6.6</version>
                </dependency>
                <dependency>
                    <groupId>com.fasterxml.jackson.core</groupId>
                    <artifactId>jackson-databind</artifactId>
                    <version>2.5.0</version>
                </dependency>
                </dependencies>
```

詳細については、Spring Framework ドキュメントの「STOMP Support」を参照してください。

AmazonMQExample.java

> **Important**
>
> 以下のコード例では、プロデューサーとコンシューマーが単一のスレッド内で実行されます。実稼働システム (またはブローカーインスタンスのフェイルオーバーをテストする) には、プロデューサーとコンシューマーが個別のホストまたはスレッドで実行されるようにしてください。

OpenWire

```
/*
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * Licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License.
 * A copy of the License is located at
 *
 *  https://aws.amazon.com/apache2.0
 *
 * or in the "license" file accompanying this file. This file is distributed
 * on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
```

```
 * express or implied. See the License for the specific language governing
 * permissions and limitations under the License.
 *
 */

                    import org.apache.activemq.ActiveMQConnectionFactory;
                    import org.apache.activemq.jms.pool.PooledConnectionFactory;

                    import javax.jms.*;

                    public class AmazonMQExample {

                    // Specify the connection parameters.
                    private final static String WIRE_LEVEL_ENDPOINT
                            = "ssl://b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-
east-2.amazonaws.com:61617";
                    private final static String ACTIVE_MQ_USERNAME =
 "MyUsername123";
                    private final static String ACTIVE_MQ_PASSWORD =
 "MyPassword456";

                    public static void main(String[] args) throws JMSException {
                        final ActiveMQConnectionFactory connectionFactory =
                                createActiveMQConnectionFactory();
                        final PooledConnectionFactory pooledConnectionFactory =
                                createPooledConnectionFactory(connectionFactory);

                        sendMessage(pooledConnectionFactory);
                        receiveMessage(connectionFactory);

                        pooledConnectionFactory.stop();
                    }

                    private static void
                    sendMessage(PooledConnectionFactory pooledConnectionFactory)
 throws JMSException {
                        // Establish a connection for the producer.
                        final Connection producerConnection =
 pooledConnectionFactory
                                .createConnection();
                        producerConnection.start();

                        // Create a session.
                        final Session producerSession = producerConnection
```

```
                                .createSession(false, Session.AUTO_ACKNOWLEDGE);

                        // Create a queue named "MyQueue".
                        final Destination producerDestination = producerSession
                                .createQueue("MyQueue");

                        // Create a producer from the session to the queue.
                        final MessageProducer producer = producerSession
                                .createProducer(producerDestination);
                        producer.setDeliveryMode(DeliveryMode.NON_PERSISTENT);

                        // Create a message.
                        final String text = "Hello from Amazon MQ!";
                        final TextMessage producerMessage = producerSession
                                .createTextMessage(text);

                        // Send the message.
                        producer.send(producerMessage);
                        System.out.println("Message sent.");

                        // Clean up the producer.
                        producer.close();
                        producerSession.close();
                        producerConnection.close();
                    }

                    private static void
                    receiveMessage(ActiveMQConnectionFactory connectionFactory)
throws JMSException {
                        // Establish a connection for the consumer.
                        // Note: Consumers should not use PooledConnectionFactory.
                        final Connection consumerConnection =
connectionFactory.createConnection();
                        consumerConnection.start();

                        // Create a session.
                        final Session consumerSession = consumerConnection
                                .createSession(false, Session.AUTO_ACKNOWLEDGE);

                        // Create a queue named "MyQueue".
                        final Destination consumerDestination = consumerSession
                                .createQueue("MyQueue");

                        // Create a message consumer from the session to the queue.
```

```
                        final MessageConsumer consumer = consumerSession
                                .createConsumer(consumerDestination);

                        // Begin to wait for messages.
                        final Message consumerMessage = consumer.receive(1000);

                        // Receive the message when it arrives.
                        final TextMessage consumerTextMessage = (TextMessage)
consumerMessage;
                        System.out.println("Message received: " +
consumerTextMessage.getText());

                        // Clean up the consumer.
                        consumer.close();
                        consumerSession.close();
                        consumerConnection.close();
                    }

                    private static PooledConnectionFactory
                    createPooledConnectionFactory(ActiveMQConnectionFactory
connectionFactory) {
                        // Create a pooled connection factory.
                        final PooledConnectionFactory pooledConnectionFactory =
                                new PooledConnectionFactory();

pooledConnectionFactory.setConnectionFactory(connectionFactory);
                        pooledConnectionFactory.setMaxConnections(10);
                        return pooledConnectionFactory;
                    }

                    private static ActiveMQConnectionFactory
createActiveMQConnectionFactory() {
                        // Create a connection factory.
                        final ActiveMQConnectionFactory connectionFactory =
                                new ActiveMQConnectionFactory(WIRE_LEVEL_ENDPOINT);

                        // Pass the sign-in credentials.
                        connectionFactory.setUserName(ACTIVE_MQ_USERNAME);
                        connectionFactory.setPassword(ACTIVE_MQ_PASSWORD);
                        return connectionFactory;
                    }
                    }
```

MQTT

```
/*
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * Licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License.
 * A copy of the License is located at
 *
 *  https://aws.amazon.com/apache2.0
 *
 * or in the "license" file accompanying this file. This file is distributed
 * on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
 * express or implied. See the License for the specific language governing
 * permissions and limitations under the License.
 *
 */

                    import org.eclipse.paho.client.mqttv3.*;

                    public class AmazonMQExampleMqtt implements MqttCallback {

                    // Specify the connection parameters.
                    private final static String WIRE_LEVEL_ENDPOINT =
                            "ssl://b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-
east-2.amazonaws.com:8883";
                    private final static String ACTIVE_MQ_USERNAME =
 "MyUsername123";
                    private final static String ACTIVE_MQ_PASSWORD =
 "MyPassword456";

                    public static void main(String[] args) throws Exception {
                        new AmazonMQExampleMqtt().run();
                    }

                    private void run() throws MqttException, InterruptedException {

                        // Specify the topic name and the message text.
                        final String topic = "myTopic";
                        final String text = "Hello from Amazon MQ!";

                        // Create the MQTT client and specify the connection
 options.
                        final String clientId = "abc123";
```

```
                        final MqttClient client = new
MqttClient(WIRE_LEVEL_ENDPOINT, clientId);
                        final MqttConnectOptions connOpts = new
MqttConnectOptions();

                        // Pass the sign-in credentials.
                        connOpts.setUserName(ACTIVE_MQ_USERNAME);
                        connOpts.setPassword(ACTIVE_MQ_PASSWORD.toCharArray());

                        // Create a session and subscribe to a topic filter.
                        client.connect(connOpts);
                        client.setCallback(this);
                        client.subscribe("+");

                        // Create a message.
                        final MqttMessage message = new
MqttMessage(text.getBytes());

                        // Publish the message to a topic.
                        client.publish(topic, message);
                        System.out.println("Published message.");

                        // Wait for the message to be received.
                        Thread.sleep(3000L);

                        // Clean up the connection.
                        client.disconnect();
                    }

                    @Override
                    public void connectionLost(Throwable cause) {
                        System.out.println("Lost connection.");
                    }

                    @Override
                    public void messageArrived(String topic, MqttMessage message)
throws MqttException {
                        System.out.println("Received message from topic " + topic +
": " + message);
                    }

                    @Override
                    public void deliveryComplete(IMqttDeliveryToken token) {
                        System.out.println("Delivered message.");
```

```
                    }
                    }
```

STOMP+WSS

```
/*
 * Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
 *
 * Licensed under the Apache License, Version 2.0 (the "License").
 * You may not use this file except in compliance with the License.
 * A copy of the License is located at
 *
 *  https://aws.amazon.com/apache2.0
 *
 * or in the "license" file accompanying this file. This file is distributed
 * on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
 * express or implied. See the License for the specific language governing
 * permissions and limitations under the License.
 *
 */

                    import
 org.springframework.messaging.converter.StringMessageConverter;
                    import org.springframework.messaging.simp.stomp.*;
                    import org.springframework.web.socket.WebSocketHttpHeaders;
                    import org.springframework.web.socket.client.WebSocketClient;
                    import
 org.springframework.web.socket.client.standard.StandardWebSocketClient;
                    import
 org.springframework.web.socket.messaging.WebSocketStompClient;

                    import java.lang.reflect.Type;

                    public class AmazonMQExampleStompWss {

                    // Specify the connection parameters.
                    private final static String DESTINATION = "/queue";
                    private final static String WIRE_LEVEL_ENDPOINT =
                            "wss://b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-
east-2.amazonaws.com:61619";
                    private final static String ACTIVE_MQ_USERNAME =
 "MyUsername123";
```

```
                    private final static String ACTIVE_MQ_PASSWORD =
"MyPassword456";

                    public static void main(String[] args) throws Exception {
                        final AmazonMQExampleStompWss example = new
AmazonMQExampleStompWss();

                        final StompSession stompSession = example.connect();
                        System.out.println("Subscribed to a destination using
session.");
                        example.subscribeToDestination(stompSession);

                        System.out.println("Sent message to session.");
                        example.sendMessage(stompSession);
                        Thread.sleep(60000);
                    }

                    private StompSession connect() throws Exception {
                        // Create a client.
                        final WebSocketClient client = new
StandardWebSocketClient();
                        final WebSocketStompClient stompClient = new
WebSocketStompClient(client);
                        stompClient.setMessageConverter(new
StringMessageConverter());

                        final WebSocketHttpHeaders headers = new
WebSocketHttpHeaders();

                        // Create headers with authentication parameters.
                        final StompHeaders head = new StompHeaders();
                        head.add(StompHeaders.LOGIN, ACTIVE_MQ_USERNAME);
                        head.add(StompHeaders.PASSCODE, ACTIVE_MQ_PASSWORD);

                        final StompSessionHandler sessionHandler = new
MySessionHandler();

                        // Create a connection.
                        return stompClient.connect(WIRE_LEVEL_ENDPOINT, headers,
head,
                                sessionHandler).get();
                    }
```

```
                    private void subscribeToDestination(final StompSession
stompSession) {
                        stompSession.subscribe(DESTINATION, new MyFrameHandler());
                    }

                    private void sendMessage(final StompSession stompSession) {
                        stompSession.send(DESTINATION, "Hello from Amazon
MQ!".getBytes());
                    }

                    private static class MySessionHandler extends
StompSessionHandlerAdapter {
                        public void afterConnected(final StompSession stompSession,
                                                   final StompHeaders stompHeaders) {
                            System.out.println("Connected to broker.");
                        }
                    }

                    private static class MyFrameHandler implements StompFrameHandler
{
                        public Type getPayloadType(final StompHeaders headers) {
                            return String.class;
                        }

                        public void handleFrame(final StompHeaders stompHeaders,
                                                final Object message) {
                            System.out.print("Received message from topic: " +
message);
                        }
                    }
                    }
```

Amazon MQ for ActiveMQ エンジンバージョンの管理

Apache ActiveMQ は、X.Y.Z 形式のセマンティックバージョニングに従ってバージョン番号を分類します。Amazon MQ for ActiveMQ の実装では、X はメジャーバージョンを示し、Y はマイナーバージョンを表し、Z はパッチバージョン番号を示します。Amazon MQ は、メジャーバージョン番号が変更される場合に、バージョン変更がメジャーであると見なします。例えば、バージョン 5.17 から 6.0 へのアップグレードは、メジャーバージョンアップグレードと見なされます。マイナーバージョン番号またはパッチバージョン番号のみが変わる場合、バージョン変更はマイナーと見なされます。たとえば、バージョン 5.18 から 5.19 へのアップグレードは、マイナーバージョンアップグレードと

見なされます。`autoMinorVersionUpgrade` をオンにすると、Amazon MQ はブローカーを利用可能な最新のパッチバージョンにアップグレードします。

Amazon MQ for ActiveMQ では、すべてのブローカーについて、サポートされている最新のマイナーバージョンを使用することをお勧めします。ブローカーエンジンバージョンをアップグレードする手順については、「Amazon MQ ブローカーエンジンバージョンのアップグレード」を参照してください。

Amazon MQ for ActiveMQ でサポートされるエンジンバージョン

Amazon MQ バージョンサポートカレンダーは、ブローカーエンジンバージョンがサポート終了に達するタイミングを示します。あるバージョンがサポート終了に達すると、Amazon MQ は、そのバージョンのすべてのブローカーを、サポートされている次のバージョンに自動的にアップグレードします。このアップグレードは、ブローカーのスケジュールされたメンテナンスウィンドウ内で、サポート終了日から 45 日以内に行われます。

Amazon MQ は、バージョンがサポート終了に達する少なくとも 90 日前に通知を送信します。中断を防ぐために、サポート終了日より前にブローカーをアップグレードすることをお勧めします。また、サポート終了が 30 日以内に予定されているバージョンで新しいブローカーを作成することはできません。

Apache ActiveMQ のバージョン	Amazon MQ でのサポート終了
ActiveMQ 5.19 (推奨)	
ActiveMQ 5.18	
ActiveMQ 5.17	2025 年 6 月 16 日
ActiveMQ 5.16	2024 年 11 月 15 日
ActiveMQ 5.15	2024 年 9 月 16 日

新しい Amazon MQ for ActiveMQ ブローカーを作成するときは、サポートされている任意の ActiveMQ エンジンバージョンを指定できます。ブローカーの作成時にエンジンバージョン番号を指定しない場合は、Amazon MQ により、デフォルトで自動的に最新のエンジンバージョン番号が選択されます。

エンジンバージョンのアップグレード

ブローカーはいつでも、サポートされている次のメジャーバージョンまたはマイナーバージョンに手動でアップグレードできます。自動マイナーバージョンアップグレードを有効にすると、Amazon MQ はメンテナンスウィンドウ内で、サポートされている最新のパッチバージョンにブローカーをアップグレードします。

ブローカーの手動アップグレードの詳細については、「the section called “エンジンバージョンのアップグレード”」を参照してください。

サポートされているエンジンバージョンのリスト化

describe-broker-instance-options AWS CLI コマンドを使用して、サポートされているすべてのマイナーエンジンバージョンとメジャーエンジンバージョンを一覧表示できます。

```
aws mq describe-broker-instance-options
```

エンジンおよびインスタンスタイプで結果をフィルタリングするには、以下にあるように、--engine-type および --host-instance-type オプションを使用します。

```
aws mq describe-broker-instance-options --engine-type engine-type --host-instance-type instance-type
```

例えば、ActiveMQ と mq.m5.large インスタンスタイプで結果をフィルタリングするには、*engine-type* を ACTIVEMQ、*instance-type* を mq.m5.large に置き換えます。

Amazon MQ for ActiveMQ のベストプラクティス

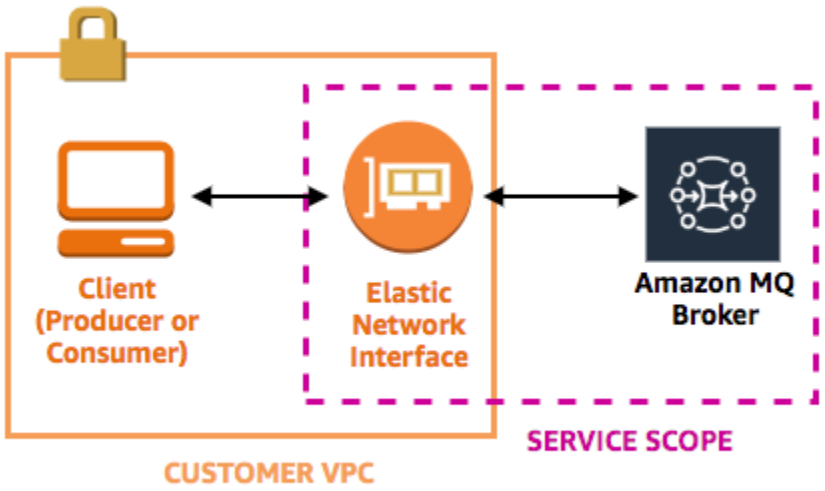
このセクションは、Amazon MQ での ActiveMQ ブローカーの使用時にパフォーマンスを最大限に引き出し、スループットコストを最小限に抑えるための推奨事項をすばやく見つけるために使用してください。

Amazon MQ Elastic Network Interface を変更または削除しない

初めて Amazon MQ ブローカーを作成するときは、Amazon MQ がアカウントの Virtual Private Cloud (VPC) 内に Elastic Network Interface をプロビジョンするため、多数の EC2 許可が必要になります。このネットワークインターフェイスは、クライアント (プロデューサーまたはコンシュー

マー) が Amazon MQ ブローカーと通信することを可能にします。このネットワークインターフェイスは、アカウントの VPC の一部であるにもかかわらず、Amazon MQ のサービス範囲内であると見なされます。

> **Warning**
>
> このネットワークインターフェイスを変更または削除しないでください。このネットワークインターフェイスを変更または削除すると、VPC とブローカーとの間の接続が完全に失われる可能性があります。



常に接続プールを使用する

単一のプロデューサーと単一のコンシューマーを使用するシナリオ (開始方法: ActiveMQ ブローカーの作成と接続チュートリアルなど) では、各プロデューサーおよびコンシューマーに単一の `ActiveMQConnectionFactory` クラスを使用できます。以下はその例です。

```
// Create a connection factory.
final ActiveMQConnectionFactory connectionFactory = new
 ActiveMQConnectionFactory(wireLevelEndpoint);

// Pass the sign-in credentials.
connectionFactory.setUserName(activeMqUsername);
connectionFactory.setPassword(activeMqPassword);
```

```
// Establish a connection for the consumer.
final Connection consumerConnection = connectionFactory.createConnection();
consumerConnection.start();
```

ただし、複数のプロデューサーやコンシューマーが関与するより現実的なシナリオでは、複数のプロデューサーのために多数の接続を作成することはコスト高および非効率的になる場合があります。このようなシナリオでは、PooledConnectionFactory クラスを使用して複数のプロデューサーリクエストをグループ化する必要があります。以下はその例です。

> **Note**
>
> メッセージコンシューマーには、PooledConnectionFactory クラスを一切使用しないでください。

```
// Create a connection factory.
final ActiveMQConnectionFactory connectionFactory = new
 ActiveMQConnectionFactory(wireLevelEndpoint);

// Pass the sign-in credentials.
connectionFactory.setUserName(activeMqUsername);
connectionFactory.setPassword(activeMqPassword);

// Create a pooled connection factory.
final PooledConnectionFactory pooledConnectionFactory = new PooledConnectionFactory();
pooledConnectionFactory.setConnectionFactory(connectionFactory);
pooledConnectionFactory.setMaxConnections(10);

// Establish a connection for the producer.
final Connection producerConnection = pooledConnectionFactory.createConnection();
producerConnection.start();
```

常にフェイルオーバートランスポートを使用して複数のブローカーエンドポイントに接続する

アクティブ/スタンバイデプロイモードを使用するとき、またはオンプレミスメッセージブローカーから Amazon MQ に移行するときなど、アプリケーションを複数のブローカーエンドポイントに接続する必要がある場合は、フェイルオーバートランスポートを使用して、コンシューマーがそれらのいずれかにランダムに接続できるようにします。例えば、次のようになります。

```
failover:(ssl://b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-
east-2.amazonaws.com:61617,ssl://b-9876l5k4-32ji-109h-8gfe-7d65c4b132a1-2.mq.us-
west-2.amazonaws.com:61617)?randomize=true
```

> **Important**
>
> マルチアベイラビリティーゾーンブローカーでは、メンテナンスウィンドウやブローカーの再起動中にフェイルオーバーが発生する可能性があります。フェイルオーバートランスポートを使用して、ブローカーの可用性を確保します。

メッセージセレクタを使用しない

JMS セレクタを使用して、トピックのサブスクリプションにフィルターをアタッチする (コンテンツに基づいてコンシューマーにメッセージを送信するため) ことは可能ですが、JMS セレクタを使用すると Amazon MQ ブローカーのフィルターバッファが満杯になり、メッセージをフィルタリングできなくなります。

一般的に、コンシューマーによるメッセージのルーティングは避けます。コンシューマーとプロデューサーが適切に非干渉化されるために、コンシューマーとプロデューサーはどちらもエフェメラルである必要があるためです。

永続サブスクリプションよりも仮想送信先を優先する

たとえば接続が失われて復元された後などに、トピックに発行されたすべてのメッセージをコンシューマーが受信するには、永続サブスクリプションが役立ちます。ただし、永続サブスクリプションを使用する場合、競合するコンシューマーの使用は不可能であり、パフォーマンスの大規模な問題が発生する可能性があります。代わりに、仮想送信先を使用することを検討してください。

Amazon VPC ピアリングを使用する場合は、CIDR 範囲 **10.0.0.0/16** 内のクライアント IP を避けてください。

オンプレミスインフラストラクチャと Amazon MQ ブローカーの間に Amazon VPC ピアリングをセットアップしている場合は、CIDR 範囲 10.0.0.0/16 内の IP でクライアント接続を設定しない必要があります。

低速コンシューマーのキューに対して同時保存とディスパッチを無効にする

デフォルトで、Amazon MQ は高速コンシューマーのキューに対して最適化を行います。

- コンシューマーは、プロデューサーによって生成されるメッセージの速度に対応できる場合、高速とみなされます。
- キューによって未確認メッセージのバックログが生成され、プロデューサーのスループットが低下する可能性がある場合、コンシューマーは低速とみなされます。

低速コンシューマーのキューに対して最適化を行うよう Amazon MQ に指示するには、`concurrentStoreAndDispatchQueues` 属性を `false` に設定します。設定の例については、「concurrentStoreAndDispatchQueues」を参照してください。

最良なスループットのために正しいブローカーインスタンスタイプを選択する

ブローカーインスタンスタイプのメッセージスループットは、アプリケーションのユースケースおよび以下の要因に依存します。

- ActiveMQ を永続モードで使用する
- メッセージサイズ
- プロデューサーとコンシューマーの数
- 送信先の数

メッセージサイズ、レイテンシー、およびスループット間の関係の理解

ユースケースによっては、より大きなブローカーインスタンスタイプはシステムスループットを向上させない場合があります。ActiveMQ が耐久性のあるストレージにメッセージを書き込むと、メッセージのサイズはシステムの制限要因を決定します。

- メッセージが 100 KB 未満の場合、永続的ストレージのレイテンシーが制限要因となります。
- メッセージが 100 KB 以上の場合、永続的ストレージのスループットが制限要因となります。

ActiveMQ を永続モード使用すると、ストレージへの書き込みは通常、前のコンシューマーがいくつか存在するか、あるいはコンシューマーが低速の場合に発生します。非永続的なモードでは、ブロー

カーインスタンスのヒープメモリに空き容量がない場合にも、低速のコンシューマーによるストレージへの書き込みが発生します。

アプリケーションにおける最適なブローカーインスタンスタイプを決定するには、異なるブローカーインスタンスタイプをテストすることが推奨されます。詳細については、「Broker instance types」および「Measuring the Throughput for Amazon MQ using the JMS Benchmark」を参照してください。

より大きなブローカーインスタンスタイプのユースケース

より大きなブローカーインスタンスタイプがスループットを向上させるには、3 つの一般的なユースケースがあります。

- 非永続モード – アプリケーションがブローカーインスタンスのフェイルオーバー中におけるメッセージの喪失による影響を受けにくいときは、多くの場合 ActiveMQ の非永続モードを使用できます。このモードでは、ブローカーインスタンスのヒープメモリに空き容量がない場合にのみ、ActiveMQ は永続的ストレージにメッセージを書き込みます。非永続モードを使用するシステムは、大きなブローカーインスタンスタイプで利用できるより大きなメモリ容量、高速の CPU、および高速のネットワークの利点を活用できます。
- 高速コンシューマー – アクティブなコンシューマーが利用可能で、`concurrentStoreAndDispatchQueues` フラグが有効になっていると、ActiveMQ は、永続モードになっている場合でも、ストレージにメッセージを送信することなく、プロデューサーからコンシューマーへの直接的なメッセージのフローを許可します。アプリケーションが素早くメッセージを消費できる場合 (あるいは、コンシューマーがその処理を行えるように設計できる場合)、アプリケーションはより大きなブローカーインスタンスタイプの利点を活用できます。アプリケーションがより素早くメッセージを消費できるようにするには、アプリケーションインスタンスにコンシューマースレッドを追加するか、あるいはアプリケーションインスタンスを水平あるいは垂直にスケールアップします。
- バッチトランザクション – 永続的モードを使用しており、トランザクションごとに複数のメッセージを送信するときは、より大きなブローカーインスタンスタイプを使用することによって、全体的に高いメッセージスループットを達成することができます。詳細については、ActiveMQ ドキュメントの「Should I Use Transactions?」を参照してください。

最高のスループットのために正しいブローカーストレージタイプを選択する

複数のアベイラビリティーゾーン全体で優れた耐障害性とレプリケーションを活用するには、Amazon EFS を使用します。低レイテンシーと高スループットを活用するには、Amazon EBS を使用します。詳細については、「Storage」を参照してください。

ブローカーのネットワークを正しく設定する

ブローカーのネットワークを作成するときは、アプリケーションに合わせて正しく設定します。

- 永続モードを有効にする – 同等のものと比べると、各ブローカーインスタンスはプロデューサーまたはコンシューマーのように動作するため、ブローカーのネットワークはメッセージの分散レプリケーションを提供しません。コンシューマーとして機能する最初のブローカーはメッセージを受信し、それをストレージに永続化します。このブローカーは確認をプロデューサーに送信し、そのメッセージを次のブローカーに転送します。2 番目のブローカーがメッセージの持続性を確認すると、最初のブローカーはそのメッセージを削除します。

  永続モードが無効になっている場合、最初のブローカーはメッセージをストレージに保持せずにプロデューサーに確認します。詳細については、Apache ActiveMQ ドキュメントの「レプリケートされたメッセージストア」および「永続的配信と非永続的配信の違い」を参照してください。
- ブローカーインスタンスのアドバイザリーメッセージを無効にしない – 詳細については、Apache ActiveMQ ドキュメントの「Advisory Message」を参照してください。
- マルチキャストブローカー検出を使用しない – Amazon MQ はマルチキャストを使用したブローカー検出をサポートしません。詳細については、Apache ActiveMQ ドキュメントの「検出、マルチキャスト、および zeroconf の違い」を参照してください。

準備された XA トランザクションを復旧することで再起動が遅くならないようにする

ActiveMQ は分散型 (XA) トランザクションをサポートしています。ActiveMQ が XA トランザクションを処理する方法を理解しておくと、Amazon MQ でのブローカーの再起動とフェイルオーバーにかかる長い復旧時間の回避に役立ちます。

未解決の準備済み XA トランザクションは、再起動のたびに再実行されます。これらのトランザクションが未解決のままである場合、その数は時間の経過とともに大きくなり、ブローカーの起動に必要な時間が大幅に長くなります。これにより、再起動とフェイルオーバー時間に影響がありま

す。commit() および rollback() を使用してこれらのトランザクションを解決し、時間の経過とともにパフォーマンスが低下しないようにする必要があります。

未解決の準備された XA トランザクションをモニタリングするには、Amazon CloudWatch Logs の JournalFilesForFastRecovery メトリクスを使用できます。この数値が増えるか、常に 1 より高い場合は、次の例のようなコードを使用して、未解決のトランザクションを復旧します。詳細については、「Amazon MQ のクォータ」を参照してください。

以下のコード例は、準備された XA トランザクションを確認し、rollback() でそれらを終了します。

```
import org.apache.activemq.ActiveMQXAConnectionFactory;

import javax.jms.XAConnection;
import javax.jms.XASession;
import javax.transaction.xa.XAResource;
import javax.transaction.xa.Xid;

public class RecoverXaTransactions {
    private static final ActiveMQXAConnectionFactory ACTIVE_MQ_CONNECTION_FACTORY;
    final static String WIRE_LEVEL_ENDPOINT =
            "tcp://localhost:61616";;
    static {
        final String activeMqUsername = "MyUsername123";
        final String activeMqPassword = "MyPassword456";
        ACTIVE_MQ_CONNECTION_FACTORY = new
 ActiveMQXAConnectionFactory(activeMqUsername, activeMqPassword, WIRE_LEVEL_ENDPOINT);
        ACTIVE_MQ_CONNECTION_FACTORY.setUserName(activeMqUsername);
        ACTIVE_MQ_CONNECTION_FACTORY.setPassword(activeMqPassword);
    }

    public static void main(String[] args) {
        try {
            final XAConnection connection =
 ACTIVE_MQ_CONNECTION_FACTORY.createXAConnection();
            XASession xaSession = connection.createXASession();
            XAResource xaRes = xaSession.getXAResource();

            for (Xid id : xaRes.recover(XAResource.TMENDRSCAN)) {
                xaRes.rollback(id);
            }
            connection.close();
```

```
        } catch (Exception e) {
        }
    }
}
```

実際のシナリオでは、XA トランザクションマネージャーに対して準備済み XA トランザクションを確認することができます。その後、rollback() または commit() を使用して準備されたトランザクションのそれぞれを処理するかどうかを決定できます。

Amazon MQ for RabbitMQ の使用

Amazon MQ は、ニーズに適したコンピューティングおよびストレージリソースを使用したメッセージブローカーの作成を容易にします。ブローカーを作成、管理 AWS マネジメントコンソール、削除するには、、Amazon MQ REST API、または を使用します AWS Command Line Interface。

このセクションでは、ActiveMQ エンジンタイプと RabbitMQ エンジンタイプ向けのメッセージブローカーの基本的要素を説明し、利用可能な Amazon MQ ブローカーのインスタンスタイプとステータスをリストして、ブローカーのアーキテクチャと設定オプションの概要を説明します。

Amazon MQ REST API については、Amazon MQ REST API リファレンスを参照してください。

Amazon MQ for RabbitMQ ブローカーとは

ブローカーは、Amazon MQ で実行されるメッセージブローカー環境です。これは、Amazon MQ の基本的な構成要素です。ブローカーインスタンスのクラス (`m7g`) およびサイズ (`large`、`medium`) を組み合わせた説明は、ブローカーインスタンスタイプ (`mq.m7g.large` など) と呼ばれます。

- 単一インスタンスブローカーは、Network Load Balancer (NLB) の背後にある 1 つのアベイラビリティーゾーンに 1 つのブローカーで構成されます。ブローカーは、アプリケーション、および Amazon EBS ストレージボリュームと通信します。
- クラスターデプロイは、ネットワークロードバランサーの内側にある 3 つの RabbitMQ ブローカーノードの論理グループで、それぞれがユーザー、キュー、および複数のアベイラビリティーゾーン (AZ) 間の分散状態を共有します。

詳細については、RabbitMQ ブローカーのデプロイ」を参照してください。

リスナーポート

Amazon MQ マネージド RabbitMQ ブローカーは、 を介したアプリケーションレベルの接続のために次のリスナーポートをサポートしていますamqps。RabbitMQ ウェブコンソールと 管理 API を使用して、クライアント接続にこれらのポートを使用することもできます。すべての接続では、セキュリティのために TLS 暗号化が使用されます。

- リスナーポート 5671 - 安全な AMQP URL を介して行われる安全な AMQP 接続に使用されます。このポートは、RabbitMQ 4 で AMQP 0-9-1 プロトコルと AMQP 1.0 プロトコルの両方をサポートしています。例えば、`us-west-2` リージョンでデプロイされた、ブロー

カー ID が b-c8352341-ec91-4a78-ad9c-a43f23d325bb のブローカーの場合、ブローカーの完全な amqps URL は b-c8352341-ec91-4a78-ad9c-a43f23d325bb.mq.us-west-2.amazonaws.com:5671 になります。

- リスナーポート443と 15671- 両方のリスナーポートを互換的に使用して、RabbitMQ ウェブコンソールまたは管理 API を介してブローカーにアクセスできます。ポート 443 は標準の HTTPS アクセスを提供しますが、ポート 15671 は TLS 暗号化を使用する従来の RabbitMQ 管理ポートです。

属性

RabbitMQ ブローカーには、いくつかの属性があります。

- 名前。例えば、MyBroker。
- ID。例えば、b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9。
- Amazon リソースネーム (ARN)。例えば、arn:aws:mq:us-east-2:123456789012:broker:MyBroker:b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9。
- RabbitMQ ウェブコンソール URL。例えば、https://b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-east-2.amazonaws.com。

  詳細については、RabbitMQ ドキュメントの「RabbitMQ web console」を参照してください。
- セキュアな AMQP エンドポイント。例えば、amqps://b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-east-2.amazonaws.com。

ブローカー属性の完全なリストについては、Amazon MQ REST API リファレンスで以下を参照してください。

- REST オペレーション ID: ブローカー
- REST オペレーション ID: ブローカー
- REST オペレーション ID: ブローカーの再起動

Amazon MQ for RabbitMQ エンジンバージョンの管理

RabbitMQ は、X.Y.Z 形式のセマンティックバージョニングに従ってバージョン番号を分類します。Amazon MQ for RabbitMQ の実装では、X はメジャーバージョンを示し、Y はマイナーバージョンを表し、Z はパッチバージョン番号を示します。Amazon MQ は、メジャーバージョン番号が変更

される場合に、バージョン変更がメジャーであると見なします。例えば、バージョン 3.13 から 4.0 へのアップグレードはメジャーバージョンのアップグレードと見なされます。マイナーバージョン番号またはパッチバージョン番号のみが変わる場合、バージョン変更はマイナーと見なされます。たとえば、バージョン 3.11.28 から 3.12.13 へのアップグレードは、マイナーバージョンアップグレードと見なされます。

Amazon MQ for RabbitMQ では、すべてのブローカーがサポートされている最新バージョンの RabbitMQ 4.2 を使用することをお勧めします。ブローカーエンジンバージョンをアップグレードする手順については、「Amazon MQ ブローカーエンジンバージョンのアップグレード」を参照してください。

新しい Amazon MQ for RabbitMQ ブローカーを作成するときは、メジャーバージョン番号とマイナーバージョン番号のみを指定する必要があります。例えば、RabbitMQ 4.2 などです。ブローカーの作成時にエンジンバージョンを指定しない場合、Amazon MQ は自動的に最新のエンジンバージョンにデフォルト設定されます。

> **Important**
>
> Amazon MQ では、ストリームはサポートされません。ストリームを作成すると、データが失われます。
> Amazon MQ は、JSON での構造化ログ記録の使用をサポートしていません。

Amazon MQ は RabbitMQ の 2 つのメジャーバージョンリリースをサポートしています。

- RabbitMQ 4

  Amazon MQ は、RabbitMQ 4 リリースシリーズの RabbitMQ 4.2 を、サポートされているすべてのインスタンスサイズにわたって mq.m7g インスタンスタイプでのみサポートしています。
- RabbitMQ 3

  Amazon MQ は、サポートされているすべてのインスタンスサイズでRabbitMQmq.t3、mq.m5、および mq.m7g インスタンスタイプの RabbitMQ 3 リリースシリーズで RabbitMQ 3.13 をサポートしています。

サポートされているエンジンバージョンのリスト化

`describe-broker-instance-options` AWS CLI コマンドを使用して、サポートされているすべてのマイナーエンジンバージョンとメジャーエンジンバージョンを一覧表示できます。

```
aws mq describe-broker-instance-options
```

エンジンおよびインスタンスタイプで結果をフィルタリングするには、以下にあるように、--engine-type および --host-instance-type オプションを使用します。

```
aws mq describe-broker-instance-options --engine-type engine-type --host-instance-type instance-type
```

例えば、ActiveMQ と mq.m7g.large インスタンスタイプで結果をフィルタリングするには、*engine-type* を RABBITMQ、*instance-type* を mq.m7g.large に置き換えます。

RabbitMQ 4

Amazon MQ は、RabbitMQ 4 リリースシリーズの RabbitMQ 4.2 を、サポートされているすべてのインスタンスサイズにわたって mq.m7g インスタンスタイプでのみサポートしています。

> **Important**
>
> RabbitMQ 4.2 では、新しいブローカーのみを作成できます。現在、RabbitMQ 3.13 からのインプレースアップグレードはサポートされていません。

> **Important**
>
> Amazon MQ for RabbitMQ 4.2 ブローカーのデフォルトのキュータイプは「クォーラム」になります。キューの作成時にキュータイプの引数を指定しない場合、クォーラムキューが作成されます。
> RabbitMQ 4 では耐久性のニーズに応じてクォーラムキューを使用することを強くお勧めします。クラシックキューは、すべてのケースで耐久性が保証されるわけではないためです。

Amazon MQ の RabbitMQ 4 では、次の変更が導入されました。

- コアプロトコルとしての AMQP 1.0: 詳細については、「プロトコル」を参照してください。
- ローカルシャベル: Shovels は、AMQP 0-9-1 および AMQP 1.0 に加えて、「local」と呼ばれる新しいプロトコルをサポートするようになりました。ローカルシャベルは内部的に AMQP 1.0 に基づいていますが、個別の TCP 接続を使用する代わりに、クラスターノードと内部 APIs 間のクラスター内接続を使用してメッセージの発行と消費を行います。これは、同じクラスター内での消費

と公開にのみ使用でき、AMQP 0-9-1 および AMQP 1.0 よりも少ないリソースを使用しながら、より高いスループットを提供できます。

- クォーラムキューはメッセージの優先順位をサポート: クォーラムキューメッセージの優先順位は常にアクティブであり、ポリシーが機能する必要はありません。クォーラムキューが優先順位が設定されたメッセージを受信するとすぐに、優先順位が有効になります。クォーラムキューは、内部的に高と通常の 2 つの優先順位のみをサポートします。優先順位が設定されていないメッセージは、優先順位 0 ~ 4 と同様に通常の にマッピングされます。優先度が 4 を超えるメッセージは高にマッピングされます。高優先度メッセージは、通常の優先度メッセージよりも 2:1 の比率で優先されます。つまり、キューは 2 つの高優先度メッセージごとに 1 つの通常の優先度メッセージ (利用可能な場合) を配信します。したがって、クォーラムキューは、厳格ではない「公平な共有」優先度処理の一種を実装します。これにより、通常の優先度メッセージで常に進行が保証されますが、高い優先度は 2:1 の比率で優先されます。
- Khepri: Khepri は RabbitMQ 4 ブローカーのデフォルトのメタデータストアとして使用されます
- 相互 TLS (mTLS): Amazon MQ は RabbitMQ ブローカーの相互 TLS (mTLS) をサポートしているため、クライアントは証明書を使用して認証できます。詳細については、mTLS 設定を参照してください。
- SSL 証明書認証プラグイン: SSL 認証プラグインは mTLS 接続のクライアント証明書を使用してユーザーを認証し、ユーザー名とパスワード認証情報の代わりに X.509 クライアント証明書を使用した認証を許可します。詳細については、「SSL 証明書認証」を参照してください。
- HTTP 認証プラグイン: HTTP 認証バックエンドプラグインを使用すると、認証と認可を外部 HTTP サービスに委任できます。詳細については、「HTTP 認証と認可」を参照してください。
- JMS サポート: ブローカーは JMS トピック交換プラグインを有効にして JMS ワークロードをサポートするようになり、JMS アプリケーションが RabbitMQ JMS クライアントを使用して接続できるようになりました。

Amazon MQ では、以下の機能が RabbitMQ 4 から廃止されました。

- クラシックキューのミラーリング: クラシックキューは、クライアントライブラリとアプリケーションに重大な変更を加えることなく引き続きサポートされますが、レプリケートされていないキュータイプになりました。クライアントは、任意のノードに接続して、レプリケートされていないクラシックキューに発行および消費できます。クォーラムキューは、レプリケーションとデータの安全性のために推奨されます。
- グローバル QoS の削除: グローバル QoS の代わりに、コンシューマーごとの QoS (非グローバル) を設定することをお勧めします。この場合、チャネル全体に単一の共有プリフェッチが使用されます。

- 一時的な非排他的キューのサポート: 一時的なキューは、宣言されているノードの稼働時間に有効期間がリンクされているキューです。単一のインスタンスブローカーでは、ノードの再起動時に削除されます。クラスターデプロイでは、ホストされているノードが再起動されると削除されます。キュー TTL は、アイドル状態の未使用キューをしばらく非アクティブ状態になった後に自動削除するために使用することをお勧めします。排他的キューは引き続きサポートされ、キューへのすべての接続が削除されると削除されます。

Amazon MQ で RabbitMQ 4.2 にアップグレードすると、以下の重大な変更がアプリケーションに影響を与える可能性があります。

- デフォルトのキュータイプ: RabbitMQ 4 ブローカーのデフォルトのキュータイプはクォーラムに設定されています。キューの作成時にキュータイプの引数を指定しない場合、クォーラムキューが作成されます。
- クォーラムキューのデフォルトの再配信制限は 20 に設定されています。20 回以上再配信されたメッセージはデッドレターまたは削除 (削除) されます。メッセージあたり 20 回の配信がキューの一般的なシナリオである場合、データ損失を避けるために、このようなキューに対してデッドレターターゲットまたは上限を設定する必要があります。これを行うには、ポリシーを使用することをお勧めします。
- amqplib: 0.10.7 より前のノード JS クライアント amqplib バージョン、または frame_max < 8192 を使用する AMQP クライアントライブラリは RabbitMQ に接続できません
- デフォルトのリソース制限: Amazon MQ for RabbitMQ では、接続、チャネル、チャネルあたりのコンシューマー、キュー、vhost、シャベル、交換、最大メッセージサイズにデフォルトのリソース使用制限が導入されました。これらはブローカーの可用性を保護するためのガードレールとして機能し、特定の要件に合わせて設定を使用してカスタマイズできます。

Amazon MQ の RabbitMQ 4 では、以下の機能はサポートされていません。

- ローカルランダム交換: Amazon MQ ノードはネットワークロードバランサーの背後にあるため、ローカルランダム交換は Amazon MQ ではサポートされていません。
- メッセージインターセプター: RabbitMQ メッセージインターセプターは Amazon MQ ではサポートされていません。
- キューごとのメトリクス: Amazon MQ は、 AWS CloudWatch を通じて RabbitMQ 4 ブローカーの RabbitMQ キューメトリクスを提供しません。Amazon MQ は引き続き AWS CloudWatch を通じてブローカーレベルのメトリクスを提供します。RabbitMQ 管理 API を使用してキューメトリクス

をクエリできます。特定のキューのメトリクスを 1 分以上の頻度でクエリすることをお勧めします。

RabbitMQ バージョンのサポート

以下の Amazon MQ バージョンサポートカレンダーは、ブローカーエンジンバージョンがサポートを終了するタイミングを示しています。あるバージョンがサポート終了に達すると、Amazon MQ は、そのバージョンのすべてのブローカーを、サポートされている次のバージョンに自動的にアップグレードします。このアップグレードは、ブローカーのスケジュールされたメンテナンス期間中に、end-of-supportから 45 日以内に行われます。

Amazon MQ は、バージョンがサポート終了に達する少なくとも 90 日前に通知を送信します。中断を防ぐために、サポート終了日より前にブローカーをアップグレードすることをお勧めします。また、サポート終了が 30 日以内に予定されているバージョンで新しいブローカーを作成することはできません。

RabbitMQ バージョン	Amazon MQ でのサポート終了
4.2 (推奨)	
3.13	
3.12	2025 年 3 月 17 日

バージョンのアップグレード

ブローカーはいつでも、サポートされている次のメジャーバージョンまたはマイナーバージョンに手動でアップグレードできます。ブローカーの手動アップグレードの詳細については、Amazon MQ ブローカーエンジンバージョンのアップグレード」を参照してください。

Amazon MQ は、バージョン 3.13 以降を使用して、すべての RabbitMQ ブローカーでサポートされている最新のパッチバージョンへのアップグレードを管理します。手動および自動のバージョンアップグレードは、どちらもスケジュールされたメンテナンスウィンドウ中、またはブローカーの再起動後に行われます。

> **Important**
>
> RabbitMQ では、バージョンの増分アップデート (例: 3.9.x から 3.10.x) のみが可能です。アップデートでマイナーバージョンをスキップすること (例: 3.8.x から 3.11.x) はできません。

再起動中、シングルインスタンスブローカーはオフラインになります。クラスターブローカーでは、ミラーキューは再起動時に同期する必要があります。キューが長い場合、キューの同期プロセスに時間がかかることがあります。キューの同期プロセス中、コンシューマーとプロデューサーはキューを利用できません。キューの同期プロセスが完了すると、ブローカーは再び利用可能になります。影響を最小限に抑えるために、トラフィックの少ない時間帯にアップグレードすることをお勧めします。バージョンアップグレードのベストプラクティスの詳細については、「Amazon MQ for RabbitMQ のベストプラクティス」を参照してください。

Amazon MQ for RabbitMQ ブローカーのデプロイオプション

RabbitMQ ブローカーは、単一インスタンスブローカーとして、またはクラスターデプロイで作成できます。どちらのデプロイモードでも、Amazon MQ はデータを冗長的に保存することによって優れた耐久性を提供します。

RabbitMQ ブローカーには、RabbitMQ がサポートする任意のプログラミング言語を使用し、以下のプロトコルに対して TLS を有効にすることによってアクセスできます。

- AMQP (0-9-1)

トピック

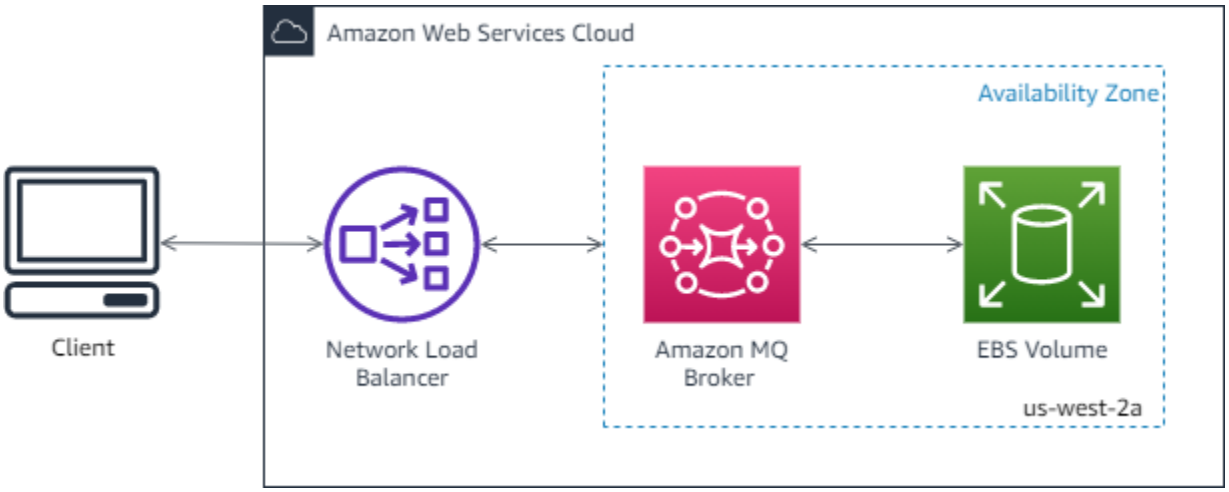
- オプション 1: Amazon MQ for RabbitMQ 単一インスタンスブローカー
- オプション 2: Amazon MQ for RabbitMQ クラスターデプロイ

オプション 1: Amazon MQ for RabbitMQ 単一インスタンスブローカー

単一インスタンスブローカーは、ネットワークロードバランサー (NLB) の内側にある 1 つのアベイラビリティーゾーン内の 1 つのブローカーで構成されます。ブローカーは、アプリケーション、および Amazon EBS ストレージボリュームと通信します。Amazon EBS は、低レイテンシーと高スループット向けに最適化されたブロックレベルのストレージを提供します。

ネットワークロードバランサーの使用は、メンテナンスウィンドウ中に、または基盤となる Amazon EC2 ハードウェア障害が理由でブローカーインスタンスが置き換えられた場合でも、Amazon MQ for RabbitMQ ブローカーエンドポイントがそのまま変更されないことを確実にします。ネットワークロードバランサーは、アプリケーションとユーザーが引き続き同じエンドポイントを使用してブローカーに接続できるようにします。

以下の図は、Amazon MQ for RabbitMQ の単一インスタンスブローカーを示しています。



オプション 2: Amazon MQ for RabbitMQ クラスターデプロイ

クラスターデプロイは、ネットワークロードバランサーの内側にある 3 つの RabbitMQ ブローカーノードの論理グループで、それぞれがユーザー、キュー、および複数のアベイラビリティーゾーン (AZ) 間の分散状態を共有します。

クラスターデプロイでは、Amazon MQ がブローカーポリシーを自動的に管理してすべてのノードでクラシックミラーリングを有効にするため、高可用性 (HA) が確保されます。ミラーされたキューはそれぞれ、1 つのメインノードと、1 つ、または複数のミラーで構成されます。各キューには独自のメインノードがあります。所定のキューに対するすべての操作は、まずキューのメインノードに適用されてから、ミラーに伝播されます。Amazon MQ は、`ha-mode` を `all`、および `ha-sync-mode` を `automatic` に設定するデフォルトのシステムポリシーを作成します。これは、より優れた耐久性のために、異なるアベイラビリティーゾーンにまたがるクラスター内のすべてのノードにデータがレプリケートされることを確実にします。

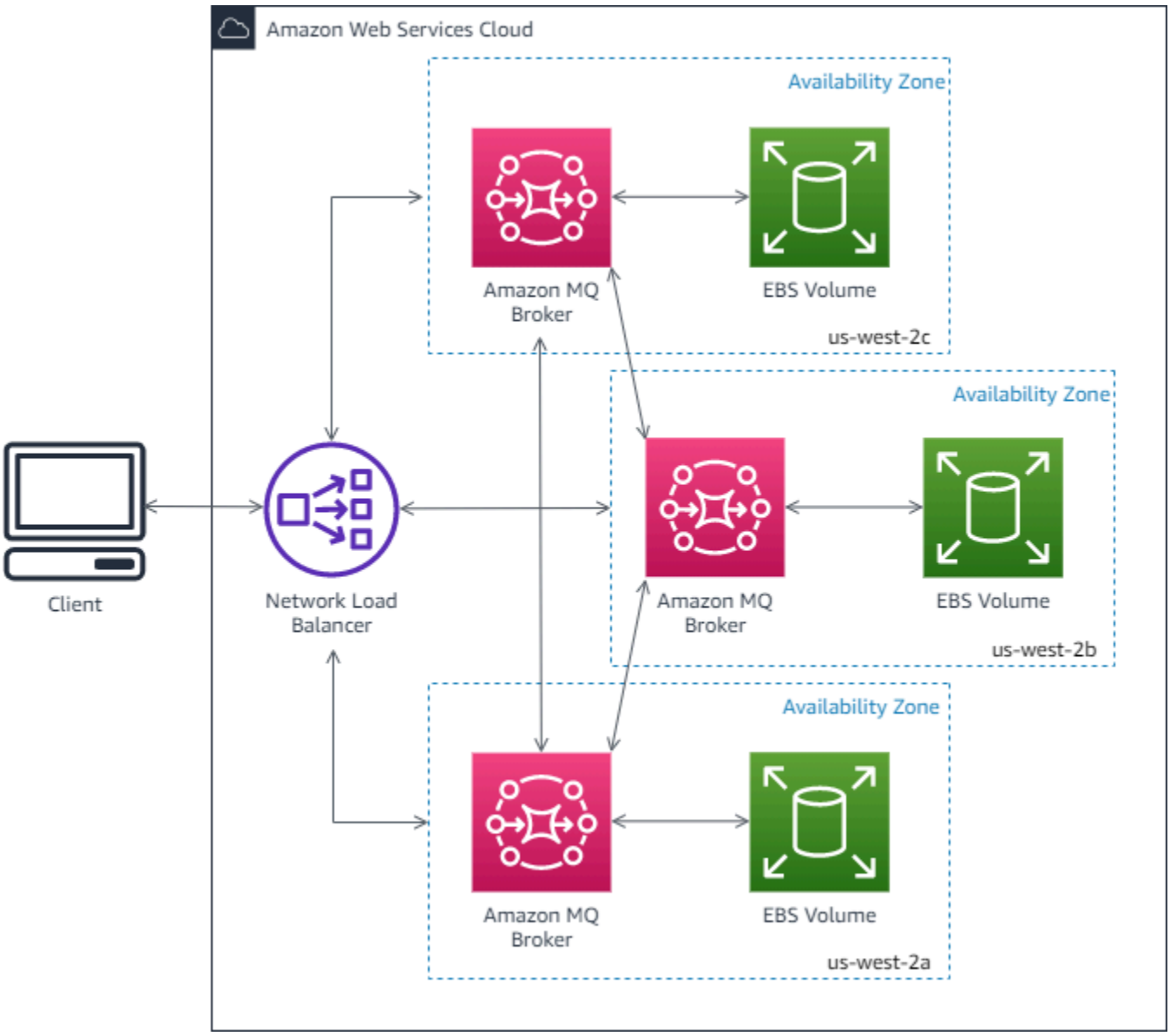
> **Note**
>
> クラスターデプロイでアベイラビリティーゾーンの停止が発生した場合、Amazon MQ はクラスターサイズを維持するために、影響を受けた RabbitMQ ノードを別の AZ に自動的に

再配置しようとします。停止が解決すると、クラスターは AZ 全体で自動的に再調整されます。

> **Note**
>
> メンテナンスウィンドウ中、クラスターに対するメンテナンスはすべて一度に 1 ノードずつ実行されるので、少なくとも 2 つのノードが常に実行され続けます。ノードへのクライアント接続は、ノードがダウンするたびに切断され、再確立されなければなりません。クライアントコードが、クラスターに自動的に再接続するように設計されていることを確認する必要があります。接続リカバリの詳細については、「the section called “ステップ 1: ネットワーク障害から自動的に回復する”」を参照してください。
>
> Amazon MQ は `ha-sync-mode: automatic` を設定するため、メンテナンスウィンドウ中、各ノードがクラスターに再参加するときにキューが同期されます。キューの同期は、その他すべてのキュー操作をブロックします。メンテナンスウィンドウ中におけるキューの同期の影響は、キューを短くしておくことによって軽減できます。

デフォルトポリシーは削除しないようにしてください。このポリシーを削除しても、Amazon MQ によって自動的に再作成されます。また、Amazon MQ は、クラスターブローカーで作成するその他すべてのポリシーに HA プロパティが適用されることも確実にします。HA プロパティのないポリシーを追加すると、Amazon MQ がそれらのプロパティを追加します。異なる高可用性プロパティを持つポリシーを追加すると、Amazon MQ がプロパティを置き換えます。クラシックミラーリングの詳細については、「Classic mirrored queues」を参照してください。

以下の図は、それぞれが独自の Amazon EBS ボリュームと共有状態を持つ 3 つのアベイラビリティーゾーン (AZ) 内に 3 つのノードがある RabbitMQ クラスターブローカーデプロイを示しています。Amazon EBS は、低レイテンシーと高スループット向けに最適化されたブロックレベルのストレージを提供します。



Amazon MQ for RabbitMQ ブローカーのインスタンスタイプ

ブローカーインスタンスクラス (m7g) とサイズ (large、medium) を組み合わせた説明は、ブローカーインスタンスタイプ (mq.m7g.large など) と呼ばれます。

クラスターデプロイと単一インスタンスデプロイの両方に mq.m7g インスタンスタイプを使用することをお勧めします。

Amazon MQ は、インスタンスタイプがサポート終了に達する少なくとも 90 日前に通知を送信します。中断を防ぐために、サポート終了日より前にブローカーを新しいインスタンスタイプにアップグレードすることをお勧めします。

> **Important**
>
> ブローカーを mq.m7gまたは mq.m5インスタンスタイプからmq.t3.microインスタンスタイプにダウングレードすることはできません。
> mq.t3.micro インスタンスタイプはクラスターのデプロイをサポートしていません。

m7g クラスターデプロイのインスタンスタイプ

クラスターデプロイで mq.m7g.x インスタンスタイプを使用することをお勧めします。次の表は、クラスターデプロイで使用可能な mq.m7g.x インスタンスタイプを示しています。

インスタンスタイプ	vCPU	メモリ (GiB)	ネットワークベースライン/バースト帯域幅 (Gbps)	推奨用途	Storage	ノードあたりのディスクボリュームサイズ (GB)
mq.m7g.medium	1	4	0.52 / 12.5	評価	EBS	5
mq.m7g.large	2	8	0.937/12.5	本番稼働	EBS	15
mq.m7g.xlarge	4	16	1.876 / 12.5	本番稼働	EBS	25
mq.m7g.2xlarge	8	32	3.75/15.0	本番稼働	EBS	45
mq.m7g.4xlarge	16	64	7.5/15.0	本番稼働	EBS	90
mq.m7g.8xlarge	32	128	15 ギガビット	本番稼働	EBS	175
mq.m7g.12xlarge	48	192	22.5 ギガビット	本番稼働	EBS	260

インスタンスタイプ	vCPU	メモリ (GiB)	ネットワークベースライン/バースト帯域幅 (Gbps)	推奨用途	Storage	ノードあたりのディスクボリュームサイズ (GB)
mq.m7g.16xlarge	64	256	30 ギガビット	本番稼働	EBS	345

m7g 単一インスタンスデプロイのインスタンスタイプ

次の表は、単一インスタンスのデプロイで使用可能な `mq.m7g.x` インスタンスタイプを示しています。

インスタンスタイプ	vCPU	メモリ (GiB)	ネットワークベースライン/バースト帯域幅 (Gbps)	推奨用途	Storage	ノードあたりのディスクボリュームサイズ (GB)
mq.m7g.medium	1	4	0.52 / 12.5	評価	EBS	200
mq.m7g.large	2	8	0.937/12.5	本番稼働	EBS	200
mq.m7g.xlarge	4	16	1.876 / 12.5	本番稼働	EBS	200
mq.m7g.2xlarge	8	32	3.75/15.0	本番稼働	EBS	200
mq.m7g.4xlarge	16	64	7.5/15.0	本番稼働	EBS	200
mq.m7g.8xlarge	32	128	15 ギガビット	本番稼働	EBS	200

インスタンスタイプ	vCPU	メモリ (GiB)	ネットワークベースライン/バースト帯域幅 (Gbps)	推奨用途	Storage	ノードあたりのディスクボリュームサイズ (GB)
mq.m7g.12xlarge	48	192	22.5 ギガビット	本番稼働	EBS	200
mq.m7g.16xlarge	64	256	39 ギガビット	本番稼働	EBS	200

mq.m5 単一インスタンスデプロイのインスタンスタイプ

次の表は、単一インスタンスデプロイで使用可能な mq.m5.x インスタンスタイプを示しています。

インスタンスタイプ	vCPU	メモリ (GiB)	ネットワークベースライン/バースト帯域幅 (Gbps)	推奨用途	Storage	ノードあたりのディスクボリュームサイズ (GB)
mq.t3.micro	2	1	0.064 / 5.0	評価	EBS	20
mq.m5.large	2	8	0.75 / 10.0	本番稼働	EBS	200
mq.m5.xlarge	4	16	1.25 / 10.0	本番稼働	EBS	200
mq.m5.2xlarge	8	32	2.5 / 10.0	本番稼働	EBS	200
mq.m5.4xlarge	16	64	5.0 / 10.0	本番稼働	EBS	200

mq.m5 クラスターデプロイのインスタンスタイプ

次の表は、クラスターデプロイで使用可能な mq.m5.x インスタンスタイプを示しています。

インスタンスタイプ	vCPU	メモリ (GiB)	ネットワークベースライン/バースト帯域幅 (Gbps)	推奨用途	Storage	ノードあたりのディスクボリュームサイズ (GB)
mq.m5.large	2	8	0.75 / 10.0	本番稼働	EBS	200
mq.m5.xlarge	4	16	1.25 / 10.0	本番稼働	EBS	200
mq.m5.2xlarge	8	32	2.5 / 10.0	本番稼働	EBS	200
mq.m5.4xlarge	16	64	5.0 / 10.0	本番稼働	EBS	200

Amazon MQ for RabbitMQ のサイズ設定ガイドライン

アプリケーションに最適なブローカーインスタンスタイプを選択できます。インスタンスタイプを選択するときは、ブローカーのパフォーマンスに影響する要因を考慮してください。

- クライアントとキューの数
- 送信されるメッセージの量
- メモリに保持されるメッセージ
- 冗長メッセージ

小規模なブローカーインスタンスタイプm7g.mediumは、アプリケーションパフォーマンスのテストにのみ推奨されます。より大きいブローカーインスタンスタイプm7g.large以上、または本番稼働レベルのクライアントとキュー、高スループット、メモリ内のメッセージ、冗長メッセージをお勧めします。

> **Important**
>
> ブローカーを mq.m5または mq.m7gインスタンスタイプから mq.t3.microインスタンスタイプにダウングレードすることはできません。

ブローカーをテストして、ワークロードのメッセージング要件に適したインスタンスタイプとサイズを決定することが重要です。

RabbitMQ 4 ブローカーのデフォルトのリソース制限を常に使用して、Amazon MQ のベストプラクティスに従ってアプリケーションに適したインスタンスサイズを決定します。これらのデフォルトのリソース制限は、m7gインスタンスタイプとクォーラムキューのタイプに基づいています。

- m7g 単一インスタンスデプロイのデフォルトのリソース制限
- m7g クラスターデプロイのデフォルトのリソース制限

任意の制限の値は、インスタンスタイプとデプロイモードによって定義された最大値まで増やすことができます。ただし、本番環境で を使用する前に、値を増やしてブローカーのパフォーマンスをテストすることを強くお勧めします。

- m7g 単一インスタンスデプロイの最大リソース制限
- m7g クラスターデプロイの最大リソース制限
- m5 単一インスタンスデプロイの最大リソース制限
- m5 クラスターデプロイの最大リソース制限
- エラーメッセージ

> **Note**
>
> RabbitMQ 3.13 ブローカーにはデフォルトのリソース制限はありませんが、推奨されるデフォルトを使用することをお勧めします。

デフォルトのリソース制限

Amazon MQ for RabbitMQ は、RabbitMQ 4 以降のブローカーリソース制限の設定をサポートしています。ブローカーを作成すると、Amazon MQ はこれらのリソース制限にデフォルト値を自動的に適

用します。これらのデフォルトは、一般的な顧客使用パターンに対応しながらブローカーの可用性を保護するためのガードレールとして機能します。特定のワークロード要件に合わせて制限設定値を変更することで、ブローカーの動作をカスタマイズできます。

変更を行う前に、次の点に注意してください。

> **Important**
>
> 1. 設定の変更はブローカーのパフォーマンスと可用性に影響を与える可能性があります
> 2. デフォルト設定オプションを変更する前に影響を理解する
> 3. 非本番環境で最初に設定変更をテストする
> 4. 変更を適用した後のブローカーのヘルスをモニタリングする

> **Important**
>
> これらの設定でサポートされているデフォルト値と範囲は、RabbitMQ バージョン、インスタンスタイプ、ブローカーデプロイモードによって異なります。

> **Important**
>
> 注: ブローカーをサポートされている範囲外の設定値に関連付けたり作成したりすると、エラーレスポンスが発生します。

RabbitMQ 4.2 ブローカーに適用されるデフォルトのリソース制限は次のとおりです。

- m7g 単一インスタンスデプロイのデフォルトのリソース制限
- m7g クラスターデプロイのデフォルトのリソース制限

デフォルトのリソース制限

> **Important**
>
> Amazon MQ for RabbitMQ 3 ブローカーの場合、デフォルトは最大リソース制限で設定され、Amazon MQ はリソース制限設定を上書きする機能を提供しません。

単一インスタンスブローカーのデフォルト値

インスタンスタイプ	ノードあたりの接続数	ノードあたりのチャネル数	チャネルあたりのコンシューマー	[キュー]	vhosts	シャベル	交換	バイト単位のメッセージサイズ
mq.m7g.medium	100	500	10	500	10	30	500	524288
mq.m7g.large	1,500	4,500	10	1,000	50	50	1,000	524288
mq.m7g.xlarge	3,000	9,000	10	2,000	100	100	2,000	524288
mq.m7g.2large	6,000	18,000	10	4,000	150	200	4,000	524288
mq.m7g.4large	12,000	36,000	10	8,000	200	400	8,000	524288
mq.m7g.8large	24,000	72,000	10	16,000	250	800	16,000	524288
mq.m7g.1xlarge	36,000	108,000	10	24,000	300	1,200	24,000	524288

インスタンスタイプ	ノードあたりの接続数	ノードあたりのチャネル数	チャネルあたりのコンシューマー	[キュー]	vhosts	シャベル	交換	バイト単位のメッセージサイズ
mq.m7g.1xlarge	48,000	144,000	10	32,000	350	1,600	32,000	524288

クラスターブローカーのデフォルト値

インスタンスタイプ	ノードあたりの接続数	ノードあたりのチャネル数	チャネルあたりのコンシューマー	[キュー]	vhosts	シャベル	交換	バイト単位のメッセージサイズ
mq.m7g.medium	100	300	10	100	10	10	100	524288
mq.m7g.large	500	1500	10	1,000	50	30	1,000	524288
mq.m7g.xlarge	1,000	3000	10	2,000	100	60	2,000	524288
mq.m7g.2large	2000	6000	10	4,000	150	120	4,000	524288
mq.m7g.4large	4000	12,000	10	8,000	200	240	8,000	524288
mq.m7g.8large	8000	24,000	10	16,000	250	480	16,000	524288

インスタンスタイプ	ノードあたりの接続数	ノードあたりのチャネル数	チャネルあたりのコンシューマー	[キュー]	vhosts	シャベル	交換	バイト単位のメッセージサイズ
mq.m7g.1 xlarge	12000	36,000	10	24,000	300	720	24000	524288
mq.m7g.1 xlarge	16,000	48,000	10	32,000	350	960	32,000	524288

Amazon MQ for RabbitMQ の最大リソース制限

単一インスタンスデプロイのクォーラムキューを使用する m7g のサイズ設定ガイドライン

次の表は、単一インスタンスブローカーの各インスタンスタイプの上限値を示しています。

インスタンスタイプ	Connectic ns	チャンネル	チャネルあたりのコンシューマー	[キュー]	Vhost	シャベル	交換	バイト単位のメッセージサイズ
mq.m7g.n dium	300	900	1,000	2,500	10	150	12500	134217728
mq.m7g.l arge	5,000	15,000	1,000	20,000	1500	250	100,000	134217728
mq.m7g.x arge	10,000	30,000	1,000	30,000	1,500	500	150,000	134217728

インスタンスタイプ	Connections	チャンネル	チャネルあたりのコンシューマー	[キュー]	Vhost	シャベル	交換	バイト単位のメッセージサイズ
mq.m7g.2large	20,000	60,000	1,000	40,000	1,500	1,000	200,000件の	134217728
mq.m7g.4large	40,000	120,000	1,000	60,000	1,500	2000	300,000	134217728
mq.m7g.8large	80,000	240,000	1,000	80,000	1,500	4000	400,000	134217728
mq.m7g.1xlarge	120,000	360,000	1,000	100,000	1,500	6,000	500,000	134217728
mq.m7g.1xlarge	160,000	480,000	1,000	120,000	1,500	8,000	600,000	134217728

クラスターデプロイのクォーラムキューを使用した m7g のサイズ設定ガイドライン

次の表は、クラスターブローカーの各インスタンスタイプの上限値を示しています。

インスタンスタイプ	ノードあたりの接続数	ノードあたりのチャネル数	チャネルあたりのコンシューマー	[キュー]	Vhost	シャベル	交換	バイト単位のメッセージサイズ
mq.m7g.medium	300	900	1,000	500	10	50	500	134217728

インスタンスタイプ	ノードあたりの接続数	ノードあたりのチャネル数	チャネルあたりのコンシューマー	[キュー]	Vhost	シャベル	交換	バイト単位のメッセージサイズ
mq.m7g.large	5,000	15,000	1,000	10,000	1,500	150	50,000	134217728
mq.m7g.xarge	10,000	30,000	1,000	15,000	1,500	300	75,000	134217728
mq.m7g.2large	20,000	60,000	1,000	20,000	1,500	600	100,000	134217728
mq.m7g.4large	40,000	120,000	1,000	30,000	1,500	1200	150,000	134217728
mq.m7g.8large	80,000	240,000	1,000	40,000	1,500	2,400	200,000件の	134217728
mq.m7g.1xlarge	120,000	360,000	1,000	50,000	1,500	3,600	250,000	134217728
mq.m7g.1xlarge	160,000	480,000	1,000	60,000	1,500	4,800	300,000	134217728

M5 単一インスタンスデプロイの最大リソース制限

次の表は、単一インスタンスブローカーの各インスタンスタイプの上限値を示しています。

インスタンスタイプ	Connections	チャンネル	チャネルあたりのコンシューマー	[キュー]	Vhost	シャベル
m5.large	5,000	15,000	1,000	30,000	1500	250

インスタンスタイプ	Connections	チャンネル	チャネルあたりのコンシューマー	[キュー]	Vhost	シャベル
m5.xlarge	10,000	30,000	1,000	60,000	1500	500
m5.2xlarge	20,000	60,000	1,000	120,000	1500	1,000
m5.4xlarge	40,000	120,000	1,000	240,000	1,000	2,000

m5 クラスターデプロイの最大リソース制限

次の表は、クラスターブローカーの各インスタンスタイプの上限値を示しています。

インスタンスタイプ	[キュー]	チャネルあたりのコンシューマー	シャベル
m5.large	10,000	1,000	150
m5.xlarge	15,000	1,000	300
m5.2xlarge	20,000	1,000	600
m5.4xlarge	30,000	1,000	1200

ノードごとに以下の接続とチャネルの制限が適用されます。

インスタンスタイプ	Connections	チャンネル
m5.large	5000	15,000
m5.xlarge	10,000	30,000
m5.2xlarge	20,000	60,000
m5.4xlarge	40,000	120,000

クラスターブローカーの正確な制限値は、利用可能なノードの数と、利用可能なノード間で RabbitMQ がどのようにリソースを分散するかに応じて、示されている値よりも低くなる場合があります。制限値を超えた場合は、別のノードへの新しい接続を作成して再試行するか、インスタンスのサイズをアップグレードして最大制限を増やすことができます。

エラーメッセージ

制限を超えると、以下のエラーメッセージが返されます。すべての値は、**m7.large** の単一インスタンスの制限が基になっています。

> **Note**
>
> 以下のメッセージのエラーコードは、使用しているクライアントライブラリによって異なる場合があります。

Connection

```
ConnectionClosedByBroker 500 "NOT_ALLOWED - connection refused: node
connection limit (5000) is reached"
```

[チャネル]

```
ConnectionClosedByBroker 1500 "NOT_ALLOWED - number of channels opened on
node 'rabbit@ip-10-0-23-173.us-west-2.compute.internal' has reached the
maximum allowed limit of (15,000)"
```

コンシューマー

```
ConnectionClosedByBroker: (530, 'NOT_ALLOWED - reached maximum (1,000) of
consumers per channel')
```

メッセージの最大サイズ

```
(406, 'PRECONDITION_FAILED - message size 524289 is larger than configured
max size 524288')
```

交換

```
(406, "PRECONDITION_FAILED - cannot declare exchange 'limit_test_3' in
vhost '/': exchange limit of 10 is reached")
```

> **Note**
>
> 次のエラーメッセージは、HTTP 管理 API 形式を使用します。

[キュー]

```
{"error":"bad_request","reason":"cannot declare queue 'my_queue': queue
limit in cluster (10,000) is reached"}]
```

シャベル

```
{"error":"bad_request","reason":"Validation failed\n\ncomponent shovel is
limited to 150 per node\n"}
```

Vhost

```
{"error":"bad_request","reason":"cannot create vhost 'my_vhost': vhost
limit of 1500 is reached"}
```

Amazon MQ for RabbitMQ のブローカーデフォルト

Amazon MQ for RabbitMQ ブローカーを作成するときは、ブローカーのパフォーマンスを最適化するために、Amazon MQ がブローカーポリシーと vhost 制限のデフォルトセットを適用します。Amazon MQ が vhost 制限を適用するのは、デフォルト (/) vhost のみです。Amazon MQ は、新しく作成された vhost にデフォルトポリシーを適用しません。すべての新規および既存のブローカーに対してこれらのデフォルトを維持することが推奨されますが、これらのデフォルトはいつでも変更、上書き、または削除できます。

Amazon MQ は、Amazon MQ for RabbitMQ 3 と RabbitMQ 4 に対して異なるブローカーポリシーと vhost 制限を作成します。 RabbitMQ 相違点については、以下のサブセクションで詳しく説明します。

Amazon MQ は、ブローカーの作成時に選択されたインスタンスタイプとブローカーデプロイモードに基づいてポリシーと制限を作成します。デフォルトポリシーの名前は、以下のように、デプロイモードに従って命名されます。

Amazon MQ for RabbitMQ 3:

- 単一インスタンス – `AWS-DEFAULT-POLICY-SINGLE-INSTANCE`

- クラスターのデプロイ – AWS-DEFAULT-POLICY-CLUSTER-MULTI-AZ && AWS-DEFAULT-QUORUM-QUEUES-POLICY-CLUSTER-MULTI-AZ

Amazon MQ for RabbitMQ 4:

- 単一インスタンス – AWS-DEFAULT-POLICY-SINGLE-INSTANCE
- クラスターのデプロイ – AWS-DEFAULT-POLICY-CLUSTER && AWS-DEFAULT-QUORUM-QUEUES-POLICY-CLUSTER-MULTI-AZ

単一インスタンスブローカーの場合、Amazon MQ はポリシーの優先順位値を 0 に設定します。デフォルトの優先順位値を上書きするには、より高い優先順位値を持つ独自のカスタムポリシーを作成することができます。クラスターデプロイの場合、Amazon MQ はブローカーデフォルトに対して優先順位値を 1 に設定します。クラスター用に独自のカスタムポリシーを作成するには、1 を超える優先順位値を割り当てます。

> **Note**
>
> クラシックミラーリングと高可用性 (HA) のため、クラスターデプロイでは ha-mode および ha-sync-mode のブローカーポリシーが必要になります。これらの設定は Amazon MQ for RabbitMQ 3 にのみ適用され、RabbitMQ 4 には設定されません。
> デフォルトの AWS-DEFAULT-POLICY-CLUSTER-MULTI-AZ ポリシーを削除する場合、Amazon MQ は優先順位値が 0 の ha-all-AWS-OWNED-DO-NOT-DELETE ポリシー使用します。これは、必要な ha-mode および ha-sync-mode ポリシーが引き続き有効であることを確実にします。独自のカスタムポリシーを作成する場合、Amazon MQ はポリシー定義に ha-mode および ha-sync-mode を自動的に付加します。

トピック

- ポリシーと制限の説明
- 推奨されるデフォルト値

ポリシーと制限の説明

以下のリストには、新しく作成されたブローカーに Amazon MQ が適用するデフォルトのポリシーと制限の説明があります。max-length、max-queues、および max-connections の値は、ブ

ローカーのインスタンスタイプとデプロイモードに応じて異なります。これらの値は、推奨されるデフォルト値 セクションにリストされています。

RabbitMQ 3 ブローカーと RabbitMQ 4 ブローカーの両方の設定

- **`queue-mode: lazy`** (ポリシー) – レイジーキューを有効にします。デフォルトで、キューはメッセージのインメモリキャッシュを保持し、ブローカーがコンシューマーにメッセージを可能な限り速く配信できるようにします。これは、ブローカーのメモリが不足し、高メモリアラームが発生する原因になる場合があります。レイジーキューは、現実的な範囲でできる限り早急にメッセージをディスクに移動しようとします。つまり、通常の動作条件下では、メモリに保持されるメッセージはそれほど多くないということです。レイジーキューを使用することにより、RabbitMQ for Amazon MQ は、はるかに大きなメッセージング負荷とはるかに長いキューをサポートできます。特定のユースケースでは、レイジーキューを使用するブローカーのパフォーマンスがわずかに遅くなる可能性があることに注意してください。これは、メッセージがインメモリキャッシュから配信されるのではなく、ディスクからブローカーに移動されるためです。

  > デプロイモード
  >
  > 単一インスタンス、クラスター

- **`max-length: number-of-messages`** (ポリシー) – キュー内のメッセージ数に対する制限を設定します。クラスターデプロイでは、この制限が、ブローカーの再起動やメンテナンスウィンドウの後などにキューの同期が一時停止されることを防ぎます。

  > デプロイモード
  >
  > クラスター

- **`overflow: reject-publish`** (ポリシー) – キュー内の数が `max-length` 値に達した後、`max-length` ポリシーを持つキューが新しいメッセージを拒否するようにします。キューがオーバーフロー状態になった場合にメッセージが失われないようにするには、ブローカーにメッセージを発行するクライアントアプリケーションがパブリッシャー確認を実装する必要があります。パブリッシャー確認の実装の詳細については、RabbitMQ ウェブサイトの「Publisher Confirms」を参照してください。

  > デプロイモード
  >
  > クラスター

RabbitMQ 3 に固有の設定

- **`max-queues: number-of-queues-per-vhost`** (vhost 制限) – ブローカー内のキューの数に対する制限を設定します。`max-length` ポリシー定義と同様に、クラスターデプロイ内のキュー数の制限は、ブローカーの再起動やメンテナンスウィンドウの後などにキューの同期が一時停止されることを防ぎます。キューの制限は、キューを維持するための過剰な CPU 量の使用も防ぎます。

  > デプロイモード
  >
  > 単一インスタンス、クラスター

- **`max-connections: number-of-connections-per-vhost`** (vhost 制限) – ブローカーへのクライアント接続数に対する制限を設定します。推奨される値に従って接続数を制限すると、ブローカーがメモリアラームを発し、操作を一時停止させる原因となり得るブローカーメモリの過剰な使用を防ぎます。

  > デプロイモード
  >
  > 単一インスタンス、クラスター

推奨されるデフォルト値

> Important
>
> `max-queues` と `max-connections`は、Amazon MQ for RabbitMQ 3 にのみ適用されます。

> Note
>
> `max-length` および `max-queue` のデフォルト制限は、5 kB の平均メッセージサイズに基づいてテストおよび評価されます。メッセージが 5 kB を大幅に超える場合は、`max-length` および `max-queue` 制限を調整して低くする必要があります。

以下の表には、新しく作成されたブローカーに対するデフォルト制限値がリストされています。Amazon MQ は、ブローカーのインスタンスタイプとデプロイモードに従ってこれらの値を適用します。

インスタンスタイプ	デプロイモード	max-length	max-queues	max-connections
mq.m7g.medium	単一インスタンス	該当なし	2,500	100
	クラスター	500,000	100	100
mq.m7g.large	単一インスタンス	該当なし	20,000	5,000
	クラスター	8,000,000	10,000	5,000
mq.m7g.xlarge	単一インスタンス	該当なし	30,000	10,000
	クラスター	9,000,000	15,000	10,000
mq.m7g.2xlarge	単一インスタンス	該当なし	40,000	20,000
	クラスター	10,000,000	40,000	20,000
mq.m7g.4xlarge	単一インスタンス	該当なし	60,000	40,000
	クラスター	12,000,000	30,000	40,000
mq.m7g.8xlarge	単一インスタンス	該当なし	80,000	80,000
	クラスター	20,000,000	40,000	80,000
mq.m7g.12xlarge	単一インスタンス	該当なし	100,000	120,000
	クラスター	30,000,000	20,000	120,000
mq.m7g.16xlarge	単一インスタンス	該当なし	120,000	160,000

インスタンスタイプ	デプロイモード	max-length	max-queues	max-connections
	クラスター	40,000,000	50,000	160,000

インスタンスタイプ	デプロイモード	max-length	max-queues	max-connections
t3.micro	単一インスタンス	該当なし	500	500
m5.large	単一インスタンス	該当なし	20,000	4,000
m5.large	クラスター	8,000,000	10,000	15,000
m5.xlarge	単一インスタンス	該当なし	30,000	8,000
m5.xlarge	クラスター	9,000,000	10,000	20,000
m5.2xlarge	単一インスタンス	該当なし	60,000	15,000
m5.2xlarge	クラスター	10,000,000	10,000	40,000
m5.4xlarge	単一インスタンス	該当なし	150,000	30,000
m5.4xlarge	クラスター	12,000,000	10,000	100,000

RabbitMQ ブローカーの設定

設定には、Cuttlefish 形式の RabbitMQ ブローカーのすべての設定が含まれます。設定は、ブローカーを作成する前に作成することができます。作成後、設定を 1 つ、または複数のブローカーに適用できます。

属性

ブローカー設定には複数の属性があります。次に例を示します。

- 名前 (MyConfiguration)
- ID (c-1234a5b6-78cd-901e-2fgh-3i45j6k178l9)
- Amazon リソースネーム (ARN) (arn:aws:mq:us-east-2:123456789012:configuration:c-1234a5b678cd-901e-2fgh-3i45j6k178l9)

設定属性の完全なリストについては、Amazon MQ REST API リファレンスで以下を参照してください。

- REST オペレーション ID: 設定
- REST オペレーション ID: 設定

設定のリビジョン属性の詳細なリストについては、以下を参照してください。

- REST オペレーション ID: 設定のリビジョン
- REST オペレーション ID: 設定のリビジョン

トピック

- RabbitMQ ブローカー設定の作成と適用
- Amazon MQ for RabbitMQ 設定リビジョンを編集する
- Amazon MQ での RabbitMQ の設定可能な値 Amazon MQ
- RabbitMQ 設定での ARN サポート

RabbitMQ ブローカー設定の作成と適用

configuration には、ActiveMQ ブローカーのすべての設定が Cuttlefish 形式で含まれています。設定は、ブローカーを作成する前に作成することができます。次に、設定を 1 つ以上のブローカーに適用できます。

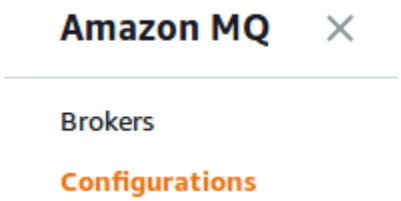
以下の例では、 AWS マネジメントコンソールを使用して Amazon MQ ブローカーの設定を作成および適用する方法を示します。

> **Important**
>
> 設定は `DeleteConfiguration` API を使用してのみ削除できます。詳細については、「Amazon MQ API リファレンス」の「設定」を参照してください。

新しい設定の作成

設定をブローカーに適用するには、まず設定を作成する必要があります。

1. Amazon MQ コンソールにサインインします。
2. 左側のナビゲーションパネルを展開し、[設定] を選択します。



3. [設定] ページで、[Create configuration (設定の作成)] を選択します。
4. [設定の作成] ページの [詳細] セクションで [設定名] (`MyConfiguration` など) を入力し、[ブローカーエンジン] のバージョンを選択します。

   Amazon MQ for ActiveMQ がサポートする RabbitMQ エンジンバージョンの詳細については、「the section called “バージョン管理”」を参照してください。
5. [設定を作成] を選択します。

新しい設定リビジョンの作成

設定を作成したら、設定リビジョンを使用して設定を編集する必要があります。

1. 設定リストから、[***`MyConfiguration`***] を選択します。

> **Note**
>
> 設定の最初のリビジョンは常に、Amazon MQ が設定を作成するときに作成されます。

[*MyConfiguration*] ページに、新しい設定リビジョンが使用するブローカーのエンジンタイプとバージョン (例: RabbitMQ 3.xx.xx) が表示されます。

2. [設定の詳細] タブに、設定リビジョン番号、説明、およびブローカー設定が Cuttlefish 形式で表示されます。

> **Note**
>
> 現在の設定を編集すると、設定の新しいリビジョンが作成されます。

3. [設定の編集] を選択して、Cuttlefish 設定を変更します。
4. [保存] を選択します。

   [Save revision] (リビジョンの保存) ダイアログボックスが表示されます。
5. (オプション) `A description of the changes in this revision` を入力します。
6. [保存] を選択します。

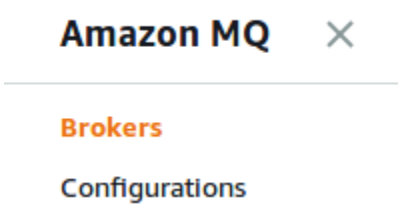
   設定の新しいリビジョンが保存されます。

> **Important**
>
> 設定を変更しても、その変更はブローカーに直ちに適用されません。変更を適用するには、次のメンテナンスウィンドウまで待機するか、ブローカーを再起動する必要があります。
> 現在、設定を削除することはできません。

設定リビジョンをブローカーに適用する

設定リビジョンを作成したら、設定リビジョンをブローカーに適用できます。

1. 左側のナビゲーションパネルを展開し、[Brokers (ブローカー)] を選択します。



2. ブローカーリストからブローカーを選択して (MyBroker など)、[Edit] (編集) をクリックします。
3. [Edit ***MyBroker***] (MyBroker の編集) ページの [Configuration] (設定) セクションで [Configuration] (設定) と [Revision] (リビジョン) を選択してから、[Schedule Modifications] (変更をスケジュールする) をクリックします。
4. [ブローカー変更のスケジュール] セクションで、変更を [次回のスケジュールされたメンテナンスウィンドウ中] に適用するか、[即時] 適用するかを選択します。

   > **Important**
   >
   > 再起動中、シングルインスタンスブローカーはオフラインになります。クラスターブローカーの場合、ブローカーの再起動中にダウンするノードは 1 つだけです。

5. [Apply] (適用) をクリックします。

   設定リビジョンが指定された時刻にブローカーに適用されます。

Amazon MQ for RabbitMQ 設定リビジョンの編集

以下の手順では、ブローカーの設定リビジョンを編集する方法について説明します。

1. Amazon MQ コンソールにサインインします。
2. ブローカーリストからブローカーを選択して (MyBroker など)、[Edit] (編集) をクリックします。
3. [***MyBroker***] ページで、[編集] を選択します。
4. [Edit ***MyBroker***] (MyBroker の編集) ページの [Configuration] (設定) セクションで [Configuration] (設定) と [Revision] (リビジョン) を選択してから、[Edit] (変更) をクリックします。

   > **Note**
   >
   > ブローカーの作成時に設定を選択する場合を除き、最初のリビジョンは、常に Amazon MQ がブローカーを作成する時に作成されます。

   [***MyBroker***] ページに、設定が使用するブローカーのエンジンタイプとバージョン (RabbitMQ 3.xx.xx など) が表示されます。
5. [設定の詳細] タブに、設定リビジョン番号、説明、およびブローカー設定が Cuttlefish 形式で表示されます。

> **Note**
>
> 現在の設定を編集すると、設定の新しいリビジョンが作成されます。

6. [設定の編集] を選択して、Cuttlefish 設定を変更します。
7. [保存] を選択します。

   [Save revision] (リビジョンの保存) ダイアログボックスが表示されます。
8. (オプション) `A description of the changes in this revision` を入力します。
9. [保存] を選択します。

   設定の新しいリビジョンが保存されます。

> **Important**
>
> 設定を変更しても、その変更はブローカーに直ちに適用されません。変更を適用するには、次のメンテナンスウィンドウまで待機するか、ブローカーを再起動する必要があります。
> 現在、設定を削除することはできません。

設定可能な値

AWS マネジメントコンソールでブローカー設定ファイルを変更することで、以下のブローカー設定オプションの値を設定できます。

次の表で説明する値に加えて、Amazon MQ は、認証と認可、およびリソース制限に関連する追加のブローカー設定オプションをサポートしています。これらの設定オプションの詳細については、「」を参照してください。

- OAuth 2.0 の設定
- LDAP 設定
- HTTP 設定
- SSL 設定
- mTLS 設定
- ARN サポート

- リソースの制限
- AMQP クライアント SSL 設定

設定	デフォルト値	推奨値	値	適用可能なバージョン	説明
consumer_timeout	1800000 ミリ秒 (30 分)	1800000 ミリ秒 (30 分)	0 ~ 2,147,483,647 ミリ秒。Amazon MQ は値 0 もサポートしています。これは「無限」を意味します。	すべてのバージョン	コンシューマーの配信承認のタイムアウト。コンシューマーが配信を受け付けない状況を検出するために使用されます。
heartbeat	60 秒	60 秒	60 ~ 3600 秒	すべてのバージョン	RabbitMQ で接続が利用できないと見なされるまでの時間を定義します。
management.restrictions.operator_policy_changes.disabled	true	true	true、false	すべてのバージョン	オペレーターポリシーへの変更を無効にします。この変更を行う場合は、HA プロパティを独自のオペレーターポリシーに含めることを強くお勧めします。

設定	デフォルト値	推奨値	値	適用可能なバージョン	説明
quorum_queue.property_equivalence.relaxed_checks_on_redeclaration	true	true	true、false	すべてのバージョン	TRUE に設定すると、アプリケーションはクォーラムキューを再宣言するときのチャネル例外を回避します。
secure.management.http.headers.enabled	true	true	true、false	すべてのバージョン	変更不可能な HTTP セキュリティヘッダーを有効にします。

コンシューマーの配信承認の設定

consumer_timeout を設定して、コンシューマーが配信を受け付けないタイミングを検出できます。コンシューマーがタイムアウト値の時間内に承認を送信しない場合、チャネルは閉じられます。例えば、デフォルト値の 1800000 ミリ秒を使用している場合は、コンシューマーが 1800000 ミリ秒以内に配信承認を送信しないとチャネルが閉じられます。Amazon MQ は値 0 もサポートしています。これは「無限」を意味します。

ハートビートの設定

ハートビートタイムアウトを設定すると、接続の中断や失敗が発生したときに検出できます。ハートビート値は、接続がダウンと見なされるまでの時間制限を定義します。

オペレーターポリシーの設定

各仮想ホストのデフォルトのオペレーターポリシーには、以下の推奨される HA プロパティがあります。

```
{
```

```
"name": "default_operator_policy_AWS_managed",
"pattern": ".*",
"apply-to": "all",
"priority": 0,
"definition": {
"ha-mode": "all",
"ha-sync-mode": "automatic"
}
}
```

AWS マネジメントコンソール または Management API を介したオペレーターポリシーの変更は、デフォルトでは利用できません。ブローカー設定に次の行を追加することで変更を有効にすることができます。

```
management.restrictions.operator_policy_changes.disabled=false
```

この変更を行う場合は、HA プロパティを独自のオペレーターポリシーに含めることを強くお勧めします。

キューの宣言時の緩和チェックの設定

クラシックキューをクォーラムキューに移行してもクライアントコードを更新していない場合は、quorum_queue.property_equivalence.relaxed_checks_on_redeclaration を true に設定することで、クォーラムキューを再宣言するときにチャネル例外を回避できます。

HTTP セキュリティヘッダーの設定

secure.management.http.headers.enabled 設定では、次の HTTP セキュリティヘッダーが有効になります。

- X-Content-Type-Options: nosniff: ブラウザによるコンテンツスニッフィングの実行を防ぎます。コンテンツスニッフィングは、ウェブサイトのファイル形式を推定するために使用されるアルゴリズムです。
- X-Frame-Options: DENY: 第三者が独自のウェブサイト上のフレームに管理プラグインを埋め込んで他者をだますことを防ぎます。
- Strict-Transport-Security: max-age=47304000; includeSubDomains: ウェブサイトとそのサブドメインに接続を行うときに、今後長期間 (1.5 年) にわたって HTTPS を使用するようにブラウザに強制します。

バージョン 3.10 以降で作成された Amazon MQ for RabbitMQ ブローカーは、デフォルトで secure.management.http.headers.enabled が true に設定されます。これらの HTTP セキュリティヘッダーを有効にするには、secure.management.http.headers.enabled を true に設定します。これらの HTTP セキュリティヘッダーをオプトアウトする場合は、secure.management.http.headers.enabled を false に設定します。

OAuth 2.0 認証と認可の設定

OAuth 2.0 設定オプションとブローカーの OAuth 2.0 認証の設定については、「サポートされている OAuth 2.0 設定」および「OAuth 2.0 認証と認可の使用」を参照してください。

LDAP 認証と認可の設定

LDAP 設定オプションとブローカーの LDAP 認証の設定については、「サポートされている LDAP 設定」および「」を参照してくださいLDAP 認証と認可の使用。

HTTP 認証と認可の設定

HTTP 認証設定値とブローカーの HTTP 認証の設定については、「HTTP 認証と認可」および「」を参照してくださいHTTP 認証と認可の使用。

> **Note**
>
> HTTP 認証プラグインは、Amazon MQ for RabbitMQ バージョン 4 以降でのみ使用できます。

SSL 証明書認証の設定

SSL 証明書認証の設定値とブローカーの SSL 証明書認証の設定については、「SSL 証明書認証」および「」を参照してくださいSSL 証明書認証の使用。

> **Note**
>
> SSL 証明書認証プラグインは、Amazon MQ for RabbitMQ バージョン 4 以降でのみ使用できます。

mTLS の設定

Amazon MQ for RabbitMQ は、さまざまなエンドポイントや外部サービスへの安全な接続のための相互 TLS (mTLS) をサポートしています。mTLS は、クライアントとサーバーの両方に証明書を使用した認証を要求することで、セキュリティを強化します。

> **Note**
>
> mTLS のプライベート認証機関の使用は、Amazon MQ for RabbitMQ バージョン 4 以降でのみ使用できます。

> **Important**
>
> Amazon MQ for RabbitMQ は、証明書とプライベートキーファイルに対して AWS ARNs の使用を強制します。詳細については、RabbitMQ 設定の ARN サポートを参照してください。

このページの内容

AMQP エンドポイント

AMQP エンドポイントへのクライアント接続に mTLS を設定します。これは SSL 証明書認証で使用されます。サポートされている設定については、「」を参照してくださいSSL 証明書認証。

RabbitMQ 管理プラグイン

RabbitMQ 管理インターフェイスへの接続用に mTLS を設定します。

> **Note**
>
> 厳格な mTLS は、 管理 API ではサポートされていません。

サポートされている設定

`aws.arns.management.ssl.cacertfile`

管理インターフェイスに接続するクライアント証明書を検証するための認証機関ファイル。

`management.ssl.verify`

ピア検証モード。サポートされている値: `verify_none`、 `verify_peer`

`management.ssl.depth`

検証のための証明書チェーンの最大深度。

`management.ssl.hostname_verification`

ホスト名検証モード。サポートされている値: `wildcard`、 `none`

サポートされていない SSL オプション

以下の SSL 設定値はサポートされていません。

完全なリストを表示する

- `management.ssl.cert`
- `management.ssl.client_renegotiation`
- `management.ssl.dh`
- `management.ssl.dhfile`
- `management.ssl.fail_if_no_peer_cert`
- `management.ssl.honor_cipher_order`
- `management.ssl.honor_ecc_order`
- `management.ssl.key.RSAPrivateKey`
- `management.ssl.key.DSAPrivateKey`
- `management.ssl.key.PrivateKeyInfo`
- `management.ssl.log_alert`

- `management.ssl.password`
- `management.ssl.psk_identity`
- `management.ssl.reuse_sessions`
- `management.ssl.secure_renegotiate`
- `management.ssl.versions.$version`
- `management.ssl.sni`

RabbitMQ OAuth 2.0 プラグイン

Amazon MQ から OAuth 2.0 ID プロバイダーへの接続用に mTLS を設定します。サポートされている設定については、「」を参照してくださいOAuth 2.0 の認証と認可。

RabbitMQ HTTP 認証プラグイン

Amazon MQ から HTTP 認証サーバーへの接続に mTLS を設定します。サポートされている設定については、「」を参照してくださいHTTP 認証と認可。

RabbitMQ LDAP プラグイン

Amazon MQ から LDAP サーバーへの接続用に mTLS を設定します。サポートされている設定については、「」を参照してくださいLDAP 認証と認可。

AMQP クライアント接続

フェデレーションとシャベルで使用される AMQP クライアント接続の TLS ピア検証を設定します。詳細については、「AMQP クライアント SSL 設定」を参照してください。

> **Important**
>
> Amazon MQ は現在、AMQP クライアント接続のクライアント証明書の設定をサポートしていません。そのため、フェデレーションとシャベルは、クライアント証明書認証を必要とする mTLS 対応ブローカーに接続できません。

リソース制限の設定

Amazon MQ for RabbitMQ は、RabbitMQ 4 以降のブローカーリソース制限の設定をサポートしています。ブローカーを作成すると、Amazon MQ はこれらのリソース制限にデフォルト値を自動的に適

用します。これらのデフォルトは、一般的な顧客使用パターンに対応しながらブローカーの可用性を保護するためのガードレールとして機能します。特定のワークロード要件に合わせて制限設定値を変更することで、ブローカーの動作をカスタマイズできます。デフォルトと最大許容値の詳細については、「」を参照してくださいthe section called “サイズ設定ガイドライン”。

リソース名と設定キー

リソース名	設定キー
接続	`connection_max`
[チャネル]	`channel_max_per_node`
[キュー]	`cluster_queue_limit`
Vhost	`vhost_max`
シャベル	`runtime_parameters.limits.shovel`
Exchange	`cluster_exchange_limit`
チャネルあたりのコンシューマー	`consumer_max_per_channel`
最大メッセージサイズ (MB)	`max_message_size`

リソース制限を上書きする方法

Amazon MQ API と Amazon MQ コンソールを使用してリソース制限を上書きできます。

次の例は、 を使用してキュー数のデフォルト制限を上書きする方法を示しています AWS CLI。

```
aws mq update-configuration --configuration-id <config-id> --data "$(echo
 "cluster_queue_limit=500" | base64 --wrap=0)"
```

呼び出しが成功すると、設定リビジョンが作成されます。設定を RabbitMQ ブローカーに関連付け、ブローカーを再起動してオーバーライドを適用する必要があります。詳細については、「」を参照してください。 RabbitMQ Broker Configurations

リソース制限オーバーライドエラー

ブローカーをサポートされている範囲外の設定値に関連付けたり作成したりすると、次のようなエラーレスポンスが発生します。

```
Configuration Revision N for configuration:cluster_queue_limit has limit: of value:
 100000000 larger than maximum allowed limit:5000
```

RabbitMQ 設定での ARN サポート

Amazon MQ for RabbitMQ は、一部の RabbitMQ 設定の値の AWS ARNs をサポートしています。これは、RabbitMQ コミュニティプラグイン rabbitmq-aws によって有効になります。このプラグインは Amazon MQ によって開発および保守されており、Amazon MQ によって管理されていないセルフホスト RabbitMQ ブローカーでも使用できます。 Amazon MQ

⚠ 重要な考慮事項

- aws プラグインによって取得された解決された ARN 値は、実行時に RabbitMQ プロセスに直接渡されます。RabbitMQ ノードの他の場所には保存されません。
- Amazon MQ for RabbitMQ には、設定された ARNs にアクセスするために Amazon MQ が引き受けることができる IAM ロールが必要です。これは、 を設定することによって設定されます`aws.arns.assume_role_arn`。
- IAM ロールを含むブローカー設定で CreateBroker または UpdateBroker APIs を呼び出すユーザーには、そのロールに対する `iam:PassRole` アクセス許可が必要です。
- IAM ロールは、RabbitMQ ブローカーと同じ AWS アカウントに存在する必要があります。設定内のすべての ARNs は、RabbitMQ ブローカーと同じ AWS リージョンに存在する必要があります。
- Amazon MQ は、IAM ロールを引き受ける`aws:SourceArn`ときに IAM グローバル条件キー`aws:SourceAccount`と を追加します。これらの値は、混乱した代理保護のためにロールにアタッチされた IAM ポリシーで使用する必要があります。

このページの内容

- IAM ポリシーのサンプル
- アクセス検証
- 関連するブローカーの隔離状態
- シナリオの例

サポートされるキー

必要な IAM ロール

`aws.arns.assume_role_arn`

Amazon MQ が他の AWS リソースにアクセスするために引き受ける IAM ロール ARN。他の ARN 設定を使用する場合に必要です。

AMQP エンドポイント

設定キー	説明
`aws.arns.ssl_options.cacertfile`	SSL/TLS クライアント接続の認証機関ファイル。Amazon MQ では、Amazon S3 または を使用して証明書を保存する必要があります。

RabbitMQ 管理プラグイン

設定キー	説明
`aws.arns.management.ssl.cacertfile`	管理インターフェイス SSL/TLS 接続の認証機関ファイル。Amazon MQ では、Amazon S3 または を使用して証明書を保存する必要があります。

RabbitMQ OAuth 2.0 プラグイン

設定キー	説明
`aws.arns.auth_oauth2.https.cacertfile`	OAuth 2.0 HTTPS 接続の認証機関ファイル。Amazon MQ では、Amazon S3 または を使用して証明書を保存する必要があります。

RabbitMQ HTTP 認証プラグイン

設定キー	説明
`aws.arns.auth_http.ssl_options.cacertfile`	HTTP 認証 SSL/TLS 接続の認証機関ファイル。Amazon MQ では、Amazon S3 または を使用して証明書を保存する必要があります。
`aws.arns.auth_http.ssl_options.certfile`	Amazon MQ と HTTP 認証サーバー間の相互 TLS 接続用の証明書ファイル。Amazon MQ では、Amazon S3 または を使用して証明書を保存する必要があります。
`aws.arns.auth_http.ssl_options.keyfile`	Amazon MQ と HTTP 認証サーバー間の相互 TLS 接続用のプライベートキーファイル。Amazon MQ では AWS Secrets Manager 、 を使用してプライベートキーを保存する必要があります。

RabbitMQ LDAP プラグイン

設定キー	説明
`aws.arns.auth_ldap.ssl_options.cacertfile`	LDAP SSL/TLS 接続の認証機関ファイル。Amazon MQ では、Amazon S3 または を使用して証明書を保存する必要があります。
`aws.arns.auth_ldap.ssl_options.certfile`	Amazon MQ と LDAP サーバー間の相互 TLS 接続用の証明書ファイル。Amazon MQ では、Amazon S3 または を使用して証明書を保存する必要があります。

設定キー	説明
`aws.arns.auth_ldap.ssl_options.keyfile`	Amazon MQ と LDAP サーバー間の相互 TLS 接続用のプライベートキーファイル。Amazon MQ では AWS Secrets Manager 、 を使用してプライベートキーを保存する必要があります。
`aws.arns.auth_ldap.dn_lookup_bind.password`	LDAP DN ルックアップバインドのパスワード。Amazon MQ では AWS Secrets Manager 、 を使用してパスワードをプレーンテキスト値として保存する必要があります。
`aws.arns.auth_ldap.other_bind.password`	LDAP の他のバインドのパスワード。Amazon MQ では AWS Secrets Manager 、 を使用してパスワードをプレーンテキスト値として保存する必要があります。

IAM ポリシーのサンプル

ロールの引き受けポリシードキュメントやロールポリシードキュメントを含む IAM ポリシーの例については、CDK サンプル実装を参照してください。

AWS Secrets Manager と Amazon S3 リソースをセットアップする手順LDAP 認証と認可の使用については、「」を参照してください。

アクセス検証

ARN 値を取得できないシナリオをトラブルシューティングするために、aws プラグインは、Amazon MQ がロールと resolveARN を正常に引き受けることができるかどうかを確認するために呼び出すことができる RabbitMQ 管理 API エンドポイントをサポートしています。 Amazon MQ AWS ARNs これにより、ブローカー設定を更新したり、新しい設定リビジョンでブローカーを更新したり、設定変更をテストするためにブローカーを再起動したりする必要がなくなります。

> **Note**
>
> この API を使用するには、既存の RabbitMQ 管理者ユーザーが必要です。Amazon MQ では、他のアクセス方法に加えて、内部ユーザーを使用してテストブローカーを作成することをお勧めします。OAuth 2.0 と簡易 (内部) 認証の両方を有効にするを参照してください。その後、このユーザーを使用して検証 API にアクセスできます。

> **Note**
>
> aws プラグインは検証 API への入力として新しいロールを渡すことをサポートしていますが、このパラメータは Amazon MQ ではサポートされていません。検証に使用する IAM ロールは、ブローカー設定`aws.arns.assume_role_arn`の の値と一致する必要があります。

関連するブローカーの隔離状態

ARN サポートの問題に関連するブローカーの隔離状態については、以下を参照してください。

- RABBITMQ_INVALID_ASSUMEROLE
- RABBITMQ_INVALID_ARN_LDAP
- RABBITMQ_INVALID_ARN

シナリオの例

- ブローカー`b-f0fc695e-2f9c-486b-845a-988023a3e55b`は、IAM ロールを使用して AWS Secrets Manager シークレット`<role>`にアクセスするように設定されています `<arn>`
- Amazon MQ に提供されたロールに AWS Secrets Manager シークレットに対する読み取りアクセス許可がない場合、RabbitMQ ログに次のエラーが表示されます。

  ```
  [error] <0.254.0> aws_arn_config: {handle_assume_role,{error,{assume_role_failed,"AWS
   service is unavailable"}}}
  ```

  さらに、ブローカーは`INVALID_ASSUMEROLE`隔離状態になります。詳細については、「INVALID_ASSUMEROLE」を参照してください。
- LDAP 認証の試行は、次のエラーで失敗します。

  ```
  [error] <0.254.0> LDAP bind failed: invalid_credentials
  ```

- 適切なアクセス許可を持つ IAM ロールを修正する
- 検証エンドポイントを呼び出して、RabbitMQ がシークレットにアクセスできるかどうかを確認します。

  ```
  curl -4su 'guest:guest' -XPUT -H 'content-type: application/json' <broker-endpoint>/
  api/aws/arn/validate -d '{"assume_role_arn":"arn:aws:iam::<account-id>:role/<role-
  ```

```
name>","arns":["arn:aws:secretsmanager:<region>:<account-id>:secret:<secret-name>"]}'
 | jq '.'
```

AMQP クライアント SSL 設定

フェデレーションとシャベルは、アップストリームブローカーとダウンストリームブローカー間の通信に AMQP を使用します。デフォルトでは、TLS ピア検証は Amazon MQ for RabbitMQ 4 の AMQP クライアントで有効になっています。この設定では、Amazon MQ ブローカーで実行されているフェデレーションおよびシャベル AMQP クライアントは、アップストリームブローカーとの接続を確立するときにピア検証を実行します。

Amazon MQ ブローカーで実行されている AMQP クライアントは、Mozilla と同じ認証機関をサポートしています。ACM を使用しない場合は、Mozilla Included CA Certificate List の CA によって発行された証明書を使用します。オンプレミスブローカーが他の認証機関の証明書を使用する場合、SSL 検証は失敗します。これらのユースケースでは、TLS ピア検証を無効にすることができます。

Important

Amazon MQ は現在、AMQP クライアント接続のクライアント証明書の設定をサポートしていません。そのため、フェデレーションとシャベルは、クライアント証明書認証を必要とする mTLS 対応ブローカーに接続できません。

Important

Amazon MQ for RabbitMQ 3 では、AMQP クライアントの SSL プロパティは RabbitMQ defaults(verify_none) で設定されます。Amazon MQ for RabbitMQ 3 では、これらのデフォルトを上書きすることはできません。

Note

`verify_peer` デフォルト設定では、任意の 2 つの Amazon MQ ブローカー間のフェデレーション接続とシャベル接続を確立できますが、Amazon MQ ブローカーとプライベートブローカー、または Amazon MQ CA 以外の証明書で実行されているオンプレミスブローカー間の接続の確立はサポートされていません。プライベートブローカーまたはオンプレミスブ

ローカーに接続するには、ダウンストリーム Amazon MQ ブローカーでピア検証を無効にする必要があります。

AMQP クライアント SSL 設定キー

設定	設定キー	サポートされる値
AMQP クライアント SSL ピア検証	`amqp_client.ssl_options.verify`	`verify_none` , `verify_peer`

AMQP クライアント SSL ピア検証を上書きする方法

RabbitMQ 4 ブローカーの Amazon MQ API と Amazon MQ コンソールを使用して、AMQP クライアントの SSL ピア検証を上書きできます。

次の例は、 を使用して AMQP クライアント SSL ピア検証を上書きする方法を示しています AWS CLI。

```
aws mq update-configuration --configuration-id <config-id> --data "$(echo
 "amqp_client.ssl_options.verify=verify_none" | base64 --wrap=0)"
```

呼び出しが成功すると、設定リビジョンが作成されます。設定を RabbitMQ ブローカーに関連付け、ブローカーを再起動してオーバーライドを適用する必要があります。詳細については、「」を参照してください。 Creating and applying broker configurations

Important

を使用する場合`verify_none`、SSL 暗号化は引き続きアクティブですが、ピアの ID は検証されません。この設定は必要な場合にのみ使用し、送信先ブローカーへのネットワークパスを信頼してください。

Amazon MQ for RabbitMQ の認証と認可

Amazon MQ for RabbitMQ は、次の認証および認可方法をサポートしています。

シンプルな認証と認可

この方法では、ブローカーユーザーは内部的に RabbitMQ ブローカーに保存され、ウェブコンソールまたは管理 API を介して管理されます。vhost、交換、キュー、トピックのアクセス許可は、RabbitMQ で直接設定されます。これがデフォルトの方法です。詳細については、「シンプルな認証と認可」を参照してください。

OAuth 2.0 の認証と認可

この方法では、ブローカーユーザーとそのアクセス許可は、外部 OAuth 2.0 ID プロバイダー (IdP) によって管理されます。vhost、交換、キュー、トピックのユーザー認証とリソースアクセス許可は、OAuth 2.0 プロバイダーのスコープシステムを通じて一元化されます。これにより、ユーザー管理が簡素化され、既存の ID システムとの統合が可能になります。詳細については、OAuth 2.0 の認証と認可」を参照してください。

IAM 認証と認可

この方法では、ブローカーユーザーは IAM AWS アウトバウンドフェデレーションを介して IAM 認証情報を使用して認証します。IAM 認証情報は、 AWS Security Token Service (STS) から JWT トークンを取得するために使用されます。これらの JWT トークンは、認証用の OAuth 2.0 トークンとして機能します。このメソッドは、Amazon MQ for RabbitMQ の既存の OAuth 2.0 サポートを活用します。 は OAuth 2.0 ID プロバイダー AWS として機能します。ユーザー認証は IAM AWS によって処理され、vhost、エクスチェンジ、キュー、トピックのリソースアクセス許可は RabbitMQ で設定された IAM ポリシーとスコープエイリアスによって管理されます。詳細については、「IAM 認証と認可」を参照してください。

LDAP 認証と認可

この方法では、ブローカーユーザーとそのアクセス許可は外部 LDAP ディレクトリサービスによって管理されます。ユーザー認証とリソースのアクセス許可は LDAP サーバーを通じて一元化されるため、ユーザーは既存のディレクトリサービス認証情報を使用して RabbitMQ にアクセスできます。詳細については、「LDAP 認証と認可」を参照してください。

HTTP 認証と認可

この方法では、ブローカーユーザーとそのアクセス許可は外部 HTTP サーバーによって管理されます。ユーザー認証とリソースのアクセス許可は HTTP サーバーを通じて一元化されるため、ユーザーは独自の認証および認可プロバイダーを使用して RabbitMQ にアクセスできます。この方法の詳細については、「HTTP 認証と認可」を参照してください。

SSL 証明書認証

Amazon MQ は、RabbitMQ ブローカーの相互 TLS (mTLS) をサポートしています。SSL 認証プラグインは、mTLS 接続からのクライアント証明書を使用してユーザーを認証します。この方法では、ブローカーユーザーはユーザー名とパスワードの認証情報の代わりに X.509 クライアント証明書を使用して認証されます。クライアントの証明書は信頼できる認証機関 (CA) に対して検証され、ユーザー名は共通名 (CN) やサブジェクト代替名 (SAN) などの証明書のフィールドから抽出されます。この方法は、ネットワーク経由で認証情報を送信せずに強力な認証を提供します。詳細については、「SSL 証明書認証」を参照してください。

> **Note**
>
> RabbitMQ は、同時に使用する複数の認証および認可方法をサポートしています。たとえば、OAuth 2.0 と簡易 (内部) 認証の両方を有効にできます。詳細については、OAuth 2.0 チュートリアルセクション「OAuth 2.0 と簡易 (内部) 認証の両方を有効にする」および「RabbitMQ アクセスコントロールドキュメント」を参照してください。
> Amazon MQ では、認証設定をテストするときに内部ユーザーを作成することをお勧めします。これにより、RabbitMQ 管理 API を使用してアクセス設定を検証できます。詳細については、「アクセス検証」を参照してください。

シンプルな認証と認可

Amazon MQ for RabbitMQ ブローカーのユーザー

> **Note**
>
> このトピックでは、RabbitMQ のデフォルトの内部認証および認可メカニズムを使用したブローカーユーザーの管理について説明します。サポートされているすべての認証および認可方法の詳細については、「Amazon MQ for RabbitMQ Authentication and Authorization」を参照してください。

すべての AMQP 0-9-1 クライアント接続には、関連付けられたユーザーがあります。このユーザーは認証されている必要があります。各クライアント接続は仮想ホスト (vhost) もターゲットにします。ユーザーには、この vhost に対する一連のアクセス許可が必要です。ユーザーは、vhost 内のキューとエクスチェンジに対して設定、書き込み、および読み込みを行う許可を持つことができます。接続が確立されると、ユーザー認証情報とターゲット vhost を指定します。

Amazon MQ for RabbitMQ ブローカーを初めて作成する場合、Amazon MQ は、指定されたサインイン認証情報を使用して、`administrator` タグで RabbitMQ ユーザーを作成します。その後、RabbitMQ Management API、または RabbitMQ ウェブコンソールを使用してユーザーを追加および管理することができます。また、RabbitMQ ウェブコンソールまたは Management API を使用して、ユーザーの認証情報とタグを設定または変更することもできます。

> **Note**
>
> RabbitMQ ユーザーは、Amazon MQ のユーザー API 経由で保存または表示されません。

> **Important**
>
> Amazon MQ for RabbitMQ では、ユーザー名「guest」はサポートされず、デフォルトのゲストアカウントは新しいブローカーの作成時に削除されます。ユーザーが作成した「guest」というアカウントも、Amazon MQ によって定期的に削除されます。

RabbitMQ Management API を使用して新しいユーザーを作成するには、以下の API エンドポイントとリクエストボディを使用します。*#####*と*#####*を、新しいサインイン認証情報に置き換えます。

```
PUT /api/users/username HTTP/1.1

    {"password":"password","tags":"administrator"}
```

> **Important**
>
> - 個人を特定できる情報 (PII) などの機密情報や秘匿性の高い情報はブローカーのユーザー名に追加しないでください。ブローカーユーザー名は、CloudWatch Logs を含む他の AWS のサービスからアクセスできます。ブローカーのユーザー名は、プライベートデータや機密データとして使用することを意図していません。
> - すべての管理者アカウントにアクセスできなくなった場合は、「復旧に IAM 認証を使用するためのブローカーアクセスの復旧」を参照してください。

tags キーは必須です。これは、ユーザーのタグのカンマで区切られたリストです。Amazon MQ は、administrator、management、monitoring、および policymaker ユーザータグをサポートします。

個々のユーザーに対する許可は、以下の API エンドポイントとリクエストボディを使用して設定できます。*vhost* および *username* を、独自の情報に置き換えます。デフォルト vhost / には、%2F を使用します。

```
PUT /api/permissions/vhost/username HTTP/1.1

    {"configure":".*","write":".*","read":".*"}
```

> **Note**
>
> configure、read、および write キーはすべて必須です。

ワイルドカード .* 値を使用することによって、このオペレーションは、指定された vhost 内のすべてのキューに対する読み取り、書き込み、および設定許可をユーザーに付与します。RabbitMQ Management API を使用したユーザーの管理の詳細については、「RabbitMQ Management HTTP API」を参照してください。

Amazon MQ for RabbitMQ に対する OAuth 2.0 の認証と認可

Amazon MQ for RabbitMQ は、複数の認証および認可方法をサポートしています。サポートされているすべての方法の詳細については、Amazon MQ for RabbitMQ ブローカーの認証と認可」を参照してください。

OAuth 2.0 の認証と認可では、ブローカーユーザーとそのアクセス許可は外部 OAuth 2.0 ID プロバイダー (IdP) によって管理されます。vhost、交換、キュー、トピックのユーザー認証とリソースアクセス許可は、OAuth 2.0 プロバイダーのスコープシステムを通じて一元化されます。これにより、ユーザー管理が簡素化され、既存の ID システムとの統合が可能になります。

> **⚠ 重要な考慮事項**
>
> - OAuth 2.0 統合は、Amazon MQ for ActiveMQ ブローカーではサポートされていません。
> - Amazon MQ for RabbitMQ では、プライベート CA によって発行されたサーバー証明書はサポートされません。

- RabbitMQ OAuth 2.0 プラグインは、トークンイントロスペクションエンドポイントと不透明なアクセストークンをサポートしていません。また、トークンの取り消しチェックも実行しません。
- 既存のブローカーで OAuth 2.0 を有効にするには、IAM アクセス許可、`mq:UpdateBrokerAccessConfiguration` を含める必要があります。
- Amazon MQ は、モニタリングのみのアクセス許可を持つ `monitoring-AWS-OWNED-DO-NOT-DELETE` という名前のシステムユーザーを自動的に作成します。このユーザーは、OAuth 2.0 対応ブローカーでも RabbitMQ の内部認証システムを使用し、ループバックインターフェイスアクセスのみに制限されています。

Amazon MQ for RabbitMQ ブローカーに OAuth 2.0 を設定する方法については、「」を参照してくださいOAuth 2.0 の認証と認可の使用。

このページの内容

サポートされている OAuth 2.0 設定

Amazon MQ for RabbitMQ は、RabbitMQ OAuth 2.0 プラグインで設定可能なすべての変数をサポートしますが、以下の例外があります。

- `auth_oauth2.https.cacertfile`
- `auth_oauth2.oauth_providers.{id/index}.https.cacertfile`
- `management.oauth_client_secret`

  Amazon MQ はこのキーをサポートしていないため、UAA は IdP としてサポートされません。
- `management.oauth_resource_servers.{id/index}.oauth_client_secret`
- `auth_oauth2.signing_keys.{id/index}`

OAuth 2.0 認証の追加検証

Amazon MQ では、OAuth 2.0 認証に次の追加の検証も適用されます。

- すべての URL は `https://` で始まる必要があります。

- サポートされている署名アルゴリズム:
  Ed25519、Ed25519ph、Ed448、Ed448ph、EdDSA、ES256K、ES256、ES384、ES512、HS256、HS3

Amazon MQ for RabbitMQ の IAM 認証と認可

Amazon MQ for RabbitMQ は、複数の認証および認可方法をサポートしています。サポートされているすべての方法の詳細については、Amazon MQ for RabbitMQ ブローカーの認証と認可」を参照してください。

IAM 認証と認可により、ブローカーユーザーは IAM AWS アウトバウンドフェデレーションを通じて IAM 認証情報を使用して認証できます。この方法では、IAM 認証情報を使用して AWS Security Token Service (STS) から JWT トークンを取得します。これらの JWT トークンは認証用の OAuth 2.0 トークンとして機能し、Amazon MQ for RabbitMQ の既存の OAuth 2.0 サポートを活用します。AWS は OAuth 2.0 ID プロバイダーとして機能します。 AWS IAM はユーザー認証を処理し、仮想ホスト、交換、キュー、トピックのリソース許可は RabbitMQ で設定された IAM ポリシーとスコープエイリアスを通じて管理されます。

⚠ 重要な考慮事項

- IAM 認証は、RabbitMQ バージョン 3.13、4.2 以降でサポートされています。Amazon MQ for ActiveMQ ブローカーではサポートされていません。
- IAM 認証では、IAM アウトバウンドフェデレーションを設定し、 AWS アカウントで使用できるようにする必要があります。
- このメソッドは、Amazon MQ for RabbitMQ の既存の OAuth 2.0 インフラストラクチャ上に構築され、OAuth 2.0 ID プロバイダー AWS として機能します。
- Amazon MQ は、モニタリングのみのアクセス許可を持つ `monitoring-AWS-OWNED-DO-NOT-DELETE` という名前のシステムユーザーを自動的に作成します。このユーザーは、IAM 対応ブローカーでも RabbitMQ の内部認証システムを使用し、ループバックインターフェイスアクセスのみに制限されています。

このページの内容

IAM 認証の仕組み

Amazon MQ for RabbitMQ の IAM 認証は、IAM アウトバウンドフェデレーションを使用して、IAM AWS 認証情報が RabbitMQ ブローカーで認証できるようにします。IAM 認証情報は、 AWS Security Token Service (STS) から JWT トークンを取得するために使用されます。これらの JWT トークンは、RabbitMQ ブローカーによる認証用の OAuth 2.0 トークンとして機能します。

制限事項

Amazon MQ for RabbitMQ の IAM 認証には、次の制限があります。

- スコープクレーム設定 – STS の JWT トークンがネストされているため、スコープクレームを直接使用することはできません。キーは でsts.amazonaws.com、RabbitMQ 設定でスコープエイリアスを使用して IAM ロールを RabbitMQ アクセス許可にマッピングする必要があります。この制限により、認可に IAM ポリシーが完全に使用されなくなり、代わりに認可に RabbitMQ 設定が必要になります。

Amazon MQ for RabbitMQ ブローカーの IAM 認証と認可を設定する方法については、「」を参照してくださいIAM 認証と認可の使用。

Amazon MQ for RabbitMQ の HTTP 認証と認可

Amazon MQ for RabbitMQ は、外部 HTTP サーバーを使用したブローカーユーザーの認証と認可をサポートしています。サポートされているその他の方法については、Amazon MQ for RabbitMQ ブローカーの認証と認可」を参照してください。

> **Note**
>
> HTTP 認証プラグインは、Amazon MQ for RabbitMQ バージョン 4 以降でのみ使用できます。

> **重要な考慮事項**
>
> - HTTP サーバーは、パブリックインターネット経由でアクセス可能である必要があります。Amazon MQ for RabbitMQ は、相互 TLS を使用して HTTP サーバーに対して認証するように設定できます。

- Amazon MQ for RabbitMQ は、ローカルファイルシステムへのアクセスを必要とする設定に AWS ARNs の使用を強制します。詳細については、RabbitMQ 設定の ARN サポートを参照してください。
- 既存のブローカーで HTTP 認証を有効にするには`mq:UpdateBrokerAccessConfiguration`、IAM アクセス許可 を含める必要があります。
- Amazon MQ は、モニタリングのみのアクセス許可を持つ `monitoring-AWS-OWNED-DO-NOT-DELETE` という名前のシステムユーザーを自動的に作成します。このユーザーは、HTTP 対応ブローカーでも RabbitMQ の内部認証システムを使用し、ループバックインターフェイスアクセスのみに制限されています。Amazon MQ は、保護されたユーザータグを追加することで、このユーザーの削除を防止します。

Amazon MQ for RabbitMQ ブローカーの HTTP 認証を設定する方法については、「」を参照してくださいHTTP 認証と認可の使用。

このページの内容

サポートされている HTTP 設定

Amazon MQ for RabbitMQ はRabbitMQ HTTP 認証プラグインですべての設定可能な変数をサポートしますが、 AWS ARNs。ARN サポートの詳細については、RabbitMQ 設定での ARN サポート」を参照してください。

ARNs を必要とする設定

`auth_http.ssl_options.cacertfile`

代わりに `aws.arns.auth_http.ssl_options.cacertfile` を使用

`auth_http.ssl_options.certfile`

代わりに `aws.arns.auth_http.ssl_options.certfile` を使用

`auth_http.ssl_options.keyfile`

代わりに `aws.arns.auth_http.ssl_options.keyfile` を使用

サポートされていない SSL オプション

以下の SSL 設定オプションもサポートされていません。

完全なリストを表示する

- auth_http.ssl_options.cert
- auth_http.ssl_options.client_renegotiation
- auth_http.ssl_options.dh
- auth_http.ssl_options.dhfile
- auth_http.ssl_options.honor_cipher_order
- auth_http.ssl_options.honor_ecc_order
- auth_http.ssl_options.key.RSAPrivateKey
- auth_http.ssl_options.key.DSAPrivateKey
- auth_http.ssl_options.key.PrivateKeyInfo
- auth_http.ssl_options.log_alert
- auth_http.ssl_options.password
- auth_http.ssl_options.psk_identity
- auth_http.ssl_options.reuse_sessions
- auth_http.ssl_options.secure_renegotiate
- auth_http.ssl_options.versions.$version
- auth_http.ssl_options.sni
- auth_http.ssl_options.crl_check

Amazon MQ での HTTP 設定の追加検証

Amazon MQ では、HTTP 認証と認可に次の追加の検証も適用されます。

- auth_http.http_method は getまたは のいずれかである必要がありますpost
- 次のパス設定では、HTTPS URLs を使用する必要があります。
  - auth_http.user_path
  - auth_http.vhost_path
  - auth_http.resource_path

- `auth_http.topic_path`
- 設定で AWS ARN の使用が必要な場合は、 を指定`aws.arns.assume_role_arn`する必要があります。

Amazon MQ for RabbitMQ の SSL 証明書認証

Amazon MQ for RabbitMQ は、X.509 クライアント証明書を使用したブローカーユーザーの認証をサポートしています。サポートされているその他の方法については、Amazon MQ for RabbitMQ ブローカーの認証と認可」を参照してください。

> **Note**
>
> SSL 証明書認証プラグインは、Amazon MQ for RabbitMQ バージョン 4 以降でのみ使用できます。

> **重要な考慮事項**
>
> - クライアント証明書は、信頼できる認証機関 (CA) によって署名されている必要があります。Amazon MQ for RabbitMQ は、認証中に証明書チェーンを検証します。
> - Amazon MQ for RabbitMQ は、CA 証明書などの証明書関連の設定や、ローカルファイルシステムへのアクセスを必要とする設定に AWS ARNs の使用を強制します。詳細については、RabbitMQ 設定の ARN サポートを参照してください。
> - Amazon MQ は、モニタリングのみのアクセス許可を持つ `monitoring-AWS-OWNED-DO-NOT-DELETE` という名前のシステムユーザーを自動的に作成します。このユーザーは、SSL 証明書が有効なブローカーでも RabbitMQ の内部認証システムを使用し、ループバックインターフェイスアクセスのみに制限されています。Amazon MQ は、保護されたユーザータグを追加することで、このユーザーの削除を防止します。

Amazon MQ for RabbitMQ ブローカーの SSL 証明書認証を設定する方法については、「」を参照してくださいSSL 証明書認証の使用。

このページの内容

サポートされている SSL 設定

Amazon MQ for RabbitMQ は、クライアント接続の SSL/TLS 設定をサポートしています。ARN サポートの詳細については、RabbitMQ 設定での ARN サポート」を参照してください。

ARNs を必要とする設定

`ssl_options.cacertfile`

代わりに `aws.arns.ssl_options.cacertfile` を使用

SSL 証明書のログイン設定

次の設定は、クライアント証明書からユーザー名を抽出する方法を制御します。

`ssl_cert_login_from`

ユーザー名抽出に使用する証明書フィールドを指定します。サポートされる値。

- `distinguished_name` - 完全な識別名を使用する
- `common_name` - Common Name (CN) フィールドを使用する
- `subject_alternative_name` または `subject_alt_name` - サブジェクトの代替名を使用する

`ssl_cert_login_san_type`

サブジェクト代替名を使用する場合、 は SAN タイプを指定します。サポートされている値: `dns`、`ip`、`email`、`uri`、 `other_name`

`ssl_cert_login_san_index`

サブジェクト代替名を使用する場合、使用する SAN エントリのインデックスを指定します (ゼロベース) 。負以外の整数である必要があります。

クライアント接続の SSL オプション

クライアント接続には、次の SSL オプションが適用されます。

`ssl_options.verify`

ピア検証モード。サポートされている値: `verify_none`、 `verify_peer`

`ssl_options.fail_if_no_peer_cert`

クライアントが証明書を提供しない場合に接続を拒否するかどうか。ブール値。

`ssl_options.depth`

検証のための証明書チェーンの最大深度。

`ssl_options.hostname_verification`

ホスト名検証モード。サポートされている値: `wildcard`、 `none`

サポートされていない SSL オプション

以下の SSL 設定オプションはサポートされていません。

完全なリストを表示する

- `ssl_options.cert`
- `ssl_options.client_renegotiation`
- `ssl_options.dh`
- `ssl_options.dhfile`
- `ssl_options.honor_cipher_order`
- `ssl_options.honor_ecc_order`
- `ssl_options.key.RSAPrivateKey`
- `ssl_options.key.DSAPrivateKey`
- `ssl_options.key.PrivateKeyInfo`
- `ssl_options.log_alert`
- `ssl_options.password`
- `ssl_options.psk_identity`
- `ssl_options.reuse_sessions`
- `ssl_options.secure_renegotiate`
- `ssl_options.versions.$version`
- `ssl_options.sni`
- `ssl_options.crl_check`

Amazon MQ での SSL 設定の追加検証

Amazon MQ では、SSL 証明書認証に次の追加の検証も適用されます。

- 設定で AWS ARN の使用が必要な場合は、 を指定`aws.arns.assume_role_arn`する必要があります。

Amazon MQ for RabbitMQ の LDAP 認証と認可

Amazon MQ for RabbitMQ は、外部 LDAP サーバーを使用したブローカーユーザーの認証と認可をサポートしています。サポートされているその他の方法については、Amazon MQ for RabbitMQ ブローカーの認証と認可」を参照してください。

⚠ 重要な考慮事項

- LDAP サーバーは、パブリックインターネット経由でアクセス可能である必要があります。Amazon MQ for RabbitMQ は、相互 TLS を使用して LDAP サーバーに対して認証するように設定できます。
- Amazon MQ for RabbitMQ は、パスワードなどの機密性の高い LDAP 設定や、ローカルファイルシステムへのアクセスを必要とする設定に AWS ARNs の使用を強制します。詳細については、RabbitMQ 設定の ARN サポートを参照してください。
- 既存のブローカーで LDAP を有効にするには`mq:UpdateBrokerAccessConfiguration`、IAM アクセス許可 を含める必要があります。
- Amazon MQ は、モニタリングのみのアクセス許可を持つ `monitoring-AWS-OWNED-DO-NOT-DELETE` という名前のシステムユーザーを自動的に作成します。このユーザーは、LDAP 対応ブローカーでも RabbitMQ の内部認証システムを使用し、ループバックインターフェイスアクセスのみに制限されています。Amazon MQ は、保護されたユーザータグを追加することで、このユーザーの削除を防止します。

Amazon MQ for RabbitMQ ブローカーの LDAP を設定する方法については、「」を参照してくださいLDAP 認証と認可の使用。

このページの内容

- Amazon MQ での LDAP 設定の追加検証

サポートされている LDAP 設定

Amazon MQ for RabbitMQ はRabbitMQ LDAP プラグインで設定可能なすべての変数をサポートしますが、 AWS ARNs。ARN サポートの詳細については、RabbitMQ 設定での ARN サポート」を参照してください。

ARNs を必要とする設定

`auth_ldap.dn_lookup_bind.password`

代わりに `aws.arns.auth_ldap.dn_lookup_bind.password` を使用

`auth_ldap.other_bind.password`

代わりに `aws.arns.auth_ldap.other_bind.password` を使用

`auth_ldap.ssl_options.cacertfile`

代わりに `aws.arns.auth_ldap.ssl_options.cacertfile` を使用

`auth_ldap.ssl_options.certfile`

代わりに `aws.arns.auth_ldap.ssl_options.certfile` を使用

`auth_ldap.ssl_options.keyfile`

代わりに `aws.arns.auth_ldap.ssl_options.keyfile` を使用

サポートされていない SSL オプション

以下の SSL 設定オプションもサポートされていません。

完全なリストを表示する

- `auth_ldap.ssl_options.cert`
- `auth_ldap.ssl_options.client_renegotiation`
- `auth_ldap.ssl_options.dh`
- `auth_ldap.ssl_options.dhfile`
- `auth_ldap.ssl_options.honor_cipher_order`

- auth_ldap.ssl_options.honor_ecc_order
- auth_ldap.ssl_options.key.RSAPrivateKey
- auth_ldap.ssl_options.key.DSAPrivateKey
- auth_ldap.ssl_options.key.PrivateKeyInfo
- auth_ldap.ssl_options.log_alert
- auth_ldap.ssl_options.password
- auth_ldap.ssl_options.psk_identity
- auth_ldap.ssl_options.reuse_sessions
- auth_ldap.ssl_options.secure_renegotiate
- auth_ldap.ssl_options.versions.$version
- auth_ldap.ssl_options.sni

Amazon MQ での LDAP 設定の追加検証

Amazon MQ では、LDAP 認証と認可に対して次の追加の検証も適用されます。

- auth_ldap.log を に設定することはできません network_unsafe
- LDAP サーバーは LDAPS を使用する必要があります。auth_ldap.use_ssl または のいずれかを明示的に有効にauth_ldap.use_starttlsする必要があります
- 設定で AWS ARN の使用が必要な場合は、 を指定aws.arns.assume_role_arnする必要があります。
- auth_ldap.servers は有効な IP アドレスまたは有効な FQDN である必要があります
- 次のキーは有効な LDAP 識別名である必要があります。
  - auth_ldap.dn_lookup_base
  - auth_ldap.dn_lookup_bind.user_dn
  - auth_ldap.other_bind.user_dn
  - auth_ldap.group_lookup_base

プラグイン

Amazon MQ for RabbitMQ は、次のプラグインもサポートしています。

- RabbitMQ 管理プラグイン

- シャベルプラグイン
- フェデレーションプラグイン
- 整合性のあるハッシュ交換プラグイン
- OAuth 2 プラグイン
- LDAP プラグイン
- HTTP プラグイン
- SSL 証明書プラグイン
- aws プラグイン
- JMS Topic Exchange プラグイン

RabbitMQ 管理プラグイン

Amazon MQ for RabbitMQ は、HTTP ベースの管理 API と RabbitMQ ウェブコンソール用のブラウザベースの UI を提供する RabbitMQ 管理プラグインをサポートしています。 RabbitMQ ブローカーのユーザーとポリシーの作成と管理には、ウェブコンソールと Management API を使用できます。

シャベルプラグイン

Amazon MQ for RabbitMQ は RabbitMQ シャベルプラグインをサポートしています。これにより、あるブローカーのキューと交換から別のブローカーにメッセージを移動できます。シャベルは、疎結合されたブローカーを接続し、メッセージ負荷が高いノードを避けてメッセージを分散するために使用できます。

> **Important**
>
> シャベル先がプライベートブローカーの場合は、キューまたはエクスチェンジの間でシャベルを構成することはできません。
> Amazon MQ は、静的シャベルの使用をサポートしません。

動的シャベルのみがサポートされています。動的シャベルはランタイムパラメータを使用して設定され、クライアント接続によっていつでもプログラムで開始および停止できます。例えば、RabbitMQ 管理 API を使用して、次の API エンドポイントへの PUT リクエストを作成して、動的シャベルを設定できます。この例では、{vhost} をブローカーの vhost の名前に置き換え、{name} を新しい動的シャベルの名前に置き換えることができます。

```
/api/parameters/shovel/{vhost}/{name}
```

リクエストボディでは、キューまたはエクスチェンジのどちらかを指定する必要がありますが、両方を指定する必要はありません。以下の例では、src-queue で指定されたローカルキューと dest-queue で定義されたリモートキューの間に動的シャベルを設定します。同様に、src-exchange パラメータと dest-exchange パラメータを使用して、2 つのエクスチェンジ間でシャベルを設定できます。

```
{
"value": {
"src-protocol": "amqp091",
"src-uri": "amqp://localhost",
"src-queue": "source-queue-name",
"dest-protocol": "amqp091",
"dest-uri": "amqps://b-c8352341-ec91-4a78-ad9c-a43f23d325bb.mq.us-
west2.amazonaws.com:5671",
"dest-queue": "destination-queue-name"
}
}
```

フェデレーションプラグイン

Amazon MQ は、RabbitMQ フェデレーションプラグインを使用してフェデレーティッドエクスチェンジとキューをサポートします。フェデレーションを使用すると、個別のブローカー上にあるキュー、エクスチェンジ、およびコンシューマー間でメッセージのフローをレプリケートできます。フェデレートされたキューとエクスチェンジは、他のブローカー内のピアへの接続にポイントツーポイントリンクを使用します。フェデレートされたエクスチェンジでは、デフォルトでメッセージが 1 回送信されますが、フェデレートされたキューでは、コンシューマーが必要とする回数だけメッセージを移動できます。

フェデレーションを使用して、アップストリームのエクスチェンジまたはキューからのメッセージをダウンストリームブローカーが消費できるようにすることが可能です。RabbitMQ ウェブコンソールまたは Management API を使用して、ダウンストリームブローカーでフェデレーションを有効にできます。

> **Important**
>
> アップストリームキューまたはエクスチェンジがプライベートブローカーにある場合は、フェデレーションを設定できません。フェデレーションは、パブリックブローカーのキュー

またはエクスチェンジの間、または、パブリックブローカーのアップストリームキューかエクスチェンジと、プライベートブローカーのダウンストリームキューかエクスチェンジの間のみ設定できます。

例えば、Management API を使用して以下を実行することにより、フェデレーションを設定できます。

- 他のノードへのフェデレーション接続を定義する 1 つ、または複数のアップストリームを設定する。フェデレーション接続は、RabbitMQ ウェブコンソールまたは Management API を使用して定義できます。管理 API を使用して、次のリクエストボディを使用して /api/parameters/federation-upstream/%2f/myupstream への POST リクエストを作成できます。

```
{"value":{"uri":"amqp://server-name","expires":3600000}}
```

- キューまたはエクスチェンジがフェデレートされるようにするポリシーを設定する。ポリシーは、RabbitMQ ウェブコンソールまたは Management API を使用して設定できます。管理 API を使用すると、次のリクエストボディを使用して /api/policies/%2f/federate-me への POST リクエストを作成できます。

```
{"pattern":"^amq\.", "definition":{"federation-upstream-set":"all"}, "apply-
to":"exchanges"}
```

> **Note**
>
> リクエストボディは、サーバー上のエクスチェンジの名前が amq で始まることを前提としています。正規表現 ^amq\ を使用します。 は、名前が「amq」で始まるすべてのエクスチェンジでフェデレーションが有効になっていることを確認します。RabbitMQ サーバー上のエクスチェンジには、異なる名前を付けることができます。

コンシステントハッシュエクスチェンジプラグイン

Amazon MQ for RabbitMQ は、RabbitMQ 整合性ハッシュ交換タイププラグインをサポートしています。コンシステントハッシュエクスチェンジは、メッセージのルーティングキーから計算されたハッシュ値に基づいてメッセージをキューに送信します。合理的に均等なルーティングキーが提供されると、コンシステントハッシュエクスチェンジはキュー間でメッセージを合理的にむらなく分散できます。

コンシステントハッシュエクスチェンジにバインドされたキューの場合、バインディングキーは各キューのバインドの重みを決定する文字列数値です。バインドの重みが高いキューでは、それらがバインドされているコンシステントハッシュエクスチェンジから受け取るメッセージの配分が相対的に高くなります。コンシステントハッシュエクスチェンジトポロジでは、パブリッシャーは単にメッセージをエクスチェンジに発行できますが、コンシューマーは特定のキューからのメッセージを消費するように明示的に設定される必要があります。

OAuth 2.0 プラグイン

Amazon MQ for RabbitMQ は、OAuth 2 認証バックエンドプラグインをサポートしています。このプラグインは、ブローカーの設定に基づいて条件付きで有効になります。有効にすると、このプラグインは OAuth 2.0 認証と認可を提供し、外部 OAuth 2.0 ID プロバイダーとの統合により、ユーザー管理とアクセスコントロールを一元化します。OAuth 2.0 認証の詳細については、「」を参照してくださいOAuth 2.0 の認証と認可。

LDAP プラグイン

Amazon MQ for RabbitMQ は、LDAP 認証バックエンドプラグインをサポートしています。このプラグインは、ブローカーの設定に基づいて条件付きで有効になります。有効にすると、このプラグインは LDAP 認証と認可を外部 LDAP ディレクトリサービスに統合して、一元化されたユーザー認証と認可を提供します。LDAP 認証の詳細については、「」を参照してくださいLDAP 認証と認可。

HTTP プラグイン

Amazon MQ for RabbitMQ は、HTTP 認証バックエンドプラグインをサポートしています。このプラグインは、ブローカーの設定に基づいて条件付きで有効になります。有効にすると、このプラグインは HTTP 認証と認可を提供し、外部 HTTP サーバーとの統合により、一元化されたユーザー認証と認可を行います。HTTP 認証の詳細については、「」を参照してくださいHTTP 認証と認可。

> **Note**
>
> HTTP 認証プラグインは、Amazon MQ for RabbitMQ バージョン 4 以降でのみ使用できます。

SSL 証明書プラグイン

Amazon MQ は、RabbitMQ ブローカーの相互 TLS (mTLS) をサポートしています。SSL 認証プラグインは、mTLS 接続のクライアント証明書を使用してユーザーを認証します。このプラグインは、ブ

ローカーの設定に基づいて条件付きで有効になります。有効にすると、ネットワーク経由で認証情報を送信せずに、X.509 クライアント証明書を使用して証明書ベースの認証を提供し、強力な認証を実現します。SSL 証明書認証の詳細については、「」を参照してくださいSSL 証明書認証。

> **Note**
>
> SSL 証明書認証プラグインは、Amazon MQ for RabbitMQ バージョン 4 以降でのみ使用できます。

aws プラグイン

aws プラグインは、ブローカーの設定に基づいて Amazon MQ for RabbitMQ によって条件付きで有効になります。Amazon MQ によって開発および保守されているこのコミュニティプラグインは、RabbitMQ 構成設定で AWS ARNs を使用して AWS のサービスから認証情報と証明書を安全に取得できます。ARN サポートの詳細については、「」を参照してくださいARN support in RabbitMQ configuration。

JMS Topic Exchange プラグイン

JMS Topic Exchange プラグインは、Amazon MQ for RabbitMQ によって常に有効になります。RabbitMQ JMS クライアントと連携して、新規および既存の JMS アプリケーションが Amazon MQ for RabbitMQ に接続できるようにします。

> **Note**
>
> JMS Topic Exchange プラグインは、Amazon MQ for RabbitMQ バージョン 4 以降でのみ使用できます。デフォルトでは有効になっていますが、RabbitMQ JMS クライアントが JMS ワークロードの実行に使用されている場合にのみアクティブ化されます。

サポートされるプロトコル

RabbitMQ ブローカーにアクセスするには、RabbitMQ がサポートする任意のプログラミング言語を使用し、次のいずれかのプロトコル仕様で TLS を有効にします。

- AMQP (0-9-1)
- AMQP 1.0

- JMS 1.1
- JMS 2.0
- JMS 3.1

Amazon MQ for RabbitMQ JMS のサポート

RabbitMQ JMS クライアントを使用して、Amazon MQ for RabbitMQ 4 で JMS 1.1、2.0RabbitMQ3.1 ワークロードを実行できるようになりました。 RabbitMQ

RabbitMQ JMS クライアント

RabbitMQ JMS クライアントは、JMS アプリケーションを Amazon MQ RabbitMQ ブローカーに接続するために必要なオープンソースの JMS クライアントライブラリです。詳細については、公式の GitHub リポジトリを参照してください。

サポートされている JMS 1.1、2.0、3.1 APIs

Amazon MQ for RabbitMQ 4 以降では、プラグイン`jms-topic-exchange`は常に有効になっています。したがって、JMS ワークロードには Amazon MQ for RabbitMQ 4 および RabbitMQ JMS クライアントを使用できます。JMS 1.1 で定義されているすべての JMS APIs は、以下を除いてサポートされています。 https://javaee.github.io/jms-spec/pages/JMS20FinalRelease#reference-implementation

- サーバーセッション APIsはサポートされていません。
- XA トランザクション APIsはサポートされていません。
- JMS キューの送信先の JMS セレクタはサポートされていません。
- JMS `NoLocal`サブスクリプション属性はサポートされていません。

JMS 2.0 および JMS 3.1 で新しく追加されたすべての APIs がサポートされています。 https://javaee.github.io/jms-spec/pages/JMS20FinalRelease#reference-implementation

- `JMSProducer.setDeliveryDelay` API はサポートされていません。

JMS アプリケーションを Amazon MQ for RabbitMQ ブローカーに接続する方法の詳細については、「JMS アプリケーションを Amazon MQ for RabbitMQ ブローカーに接続する」のチュートリアルを参照してください。

認証と認可

このセクションに記載されているすべての認証および認可メカニズムがサポートされています。JMS クライアントを使用してブローカーに接続するために使用される認証情報は、AMQP Java クライアントを使用して RabbitMQ ブローカーに接続する場合と同じです。

RabbitMQ での AMQP キューとの相互運用性

RabbitMQ JMS クライアントを使用して JMS メッセージを AMQP 交換に送信し、AMQP キューからのメッセージを消費できます (この機能は JMS トピックをサポートしていません)。これにより、特定の JMS ワークロードを AMQP ワークロードに相互運用または移行できます。詳細については、公式のクライアントドキュメントを参照してください。

Amazon MQ for RabbitMQ へのポリシーの適用

Amazon MQ の推奨デフォルト値を持つカスタムのポリシーと制限を適用できます。推奨されるデフォルトポリシーと制限を削除したが、それらを再作成したい、または追加の vhost を作成して、新しい vhost にデフォルトのポリシーと制限を適用したいという場合は、以下のステップを実行できます。

Important

Amazon MQ for RabbitMQ エンジンバージョン 3.13 以前では、現在のデフォルトのオペレータポリシーは次のとおりです。

```
vhost name pattern apply-to definition priority/
 default_operator_policy_AWS_managed .* classic_queues {"ha-mode":"all","ha-
sync-mode":"automatic","queue-version":2} 0
```

バージョン 4.0 以降では、デフォルトの演算子ポリシーが次のように変更されました。

```
vhost name pattern apply-to definition priority/
 default_operator_policy_AWS_managed .* classic_queues {"queue-version":2} 0
```

この変更は、従来のキューミラーリングと HA ポリシー設定が RabbitMQ 4 でサポートされていないために必要です。
従来のミラーキューとクォーラムキューの両方に適用されるポリシーを作成することはできません。ポリシーをクォーラムキューにのみ適用する場合は、`--apply-to` を `quorum_queues` に設定する必要があります。従来のミラーキューとクォーラムキューを使

用している場合は、クォーラムキューポリシーに加えて、`--apply-to:classic_queues` を設定した別のポリシーを作成する必要があります。

Important

以下のステップを実行するには、管理者権限を持つ Amazon MQ for RabbitMQ ブローカーユーザーが必要です。ブローカーを初めて作成したときに作成された管理者ユーザー、またはその後で作成した別のユーザーを使用できます。以下の表は、正規表現 (regexp) パターンとしての必要な管理者ユーザータグと許可です。

タグ	読み込み regexp	設定 regexp	書き込み regexp
`administrator`	`.*`	`.*`	`.*`

RabbitMQ ユーザーの作成、およびユーザータグと許可の管理の詳細については、「Amazon MQ for RabbitMQ ブローカーのユーザー」を参照してください。

RabbitMQ ウェブコンソールを使用してデフォルトのポリシーと仮想ホスト制限を適用する

1. Amazon MQ コンソールにサインインします。
2. 左側のナビゲーションペインで [Brokers] (ブローカー) をクリックします。
3. ブローカーのリストから、新しいポリシーを適用するブローカーの名前を選択します。
4. ブローカーの詳細ページの [Connections] (接続) セクションで、RabbitMQ ウェブコンソール URL をクリックします。RabbitMQ ウェブコンソールが新しいブラウザタブまたはウィンドウで開きます。
5. ブローカー管理者のユーザー名とパスワードを使用して RabbitMQ ウェブコンソールにログインします。
6. RabbitMQ ウェブコンソールのページ上部で、[Admin] (管理) をクリックします。
7. [Admin] (管理) ページの右側にあるナビゲーションペインで [Policies] (ポリシー) をクリックします。
8. [Policies] (ポリシー) ページに、ブローカーの現在の [User policies] (ユーザーポリシー) が表示されます。[User policies] (ユーザーポリシー) の下で、[Add / update a policy] (ポリシーの追加/更新) を展開します。

9. 新しいブローカーポリシーを作成するには、[Add / update a policy] (ポリシーの追加/更新) で以下を実行します。

   a. [Virtual host] (仮想ホスト) には、ドロップダウンリストからポリシーをアタッチする仮想ホストの名前を選択します。デフォルト vhost を選択するには、[/] を選択します。

   > **Note**
   >
   > 追加の vhost を作成していない場合は、RabbitMQ コンソールに [Virtual host] (仮想ホスト) オプションが表示されず、デフォルト vhost のみにポリシーが適用されます。

   b. [Name] (名前) には、ポリシーの名前 (**`policy-defaults`** など) を入力します。

   c. [Pattern] (パターン) には regexp パターン **`.*`** を入力して、ポリシーがブローカー上のすべてのキューと一致するようにします。

   d. [Apply to] (適用先) には、ドロップダウンリストから [Exchanges and queues] (エクスチェンジとキュー) を選択します。

   e. [Priority] (優先順位) には、vhost に適用されたその他すべてのポリシーよりも大きい整数を入力します。RabbitMQ のキューとエクスチェンジに適用できるのは、常に 1 つのポリシー定義セットのみです。RabbitMQ は、一致するポリシーで、最高の優先順位値を持つものを選択します。ポリシーの優先順位とポリシーの結合方法の詳細については、RabbitMQ サーバードキュメントの「Policies」を参照してください。

   f. [Definition] (定義) には、以下のキーバリューペアを追加します。

      - **`queue-mode=lazy`**。ドロップダウンリストから [String] (文字列) を選択します。
      - **`overflow=reject-publish`**。ドロップダウンリストから [String] (文字列) を選択します。

      > **Note**
      >
      > 単一インスタンスブローカーには適用されません。

      - **`max-length=`*`number-of-messages`***。*`number-of-messages`* は、ブローカーのインスタンスサイズとデプロイモードに従った Amazon MQ の推奨値 (例えば、`mq.m7g.large` クラスターには **`8000000`**) に置き換えます。ドロップダウンリストから [Number] (数値) を選択します。

> **Note**
>
> 単一インスタンスブローカーには適用されません。

g. [Add / update policy] (ポリシーを追加/更新) をクリックします。

10. [User policies] (ユーザーポリシー) リストに新しいポリシーが表示されることを確認します。

> **Note**
>
> クラスターブローカーの場合、Amazon MQ が `ha-mode: all` および `ha-sync-mode: automatic` ポリシー定義を自動的に適用します。

11. 右側のナビゲーションペインで [Limits] (制限) をクリックします。
12. [Limits] (制限) ページに、ブローカーの現在の [Virtual host limits] (仮想ホストの制限) が表示されます。[Virtual host limits] (仮想ホスト制限) で、[Set / update a virtual host limit] (仮想ホスト制限の設定/更新) を展開します。
13. 新しい vhost 制限を作成するには、[Set / update a virtual host limit] (仮想ホスト制限の設定/更新) で以下を実行します。

    a. [Virtual host] (仮想ホスト) には、ドロップダウンリストからポリシーをアタッチする仮想ホストの名前を選択します。デフォルト vhost を選択するには、[/] を選択します。

    b. [Limit] (制限) には、ドロップダウンオプションから [max-connections] を選択します。

    c. [Value] (値) には、ブローカーのインスタンスサイズとデプロイモードに従った Amazon MQ の推奨値 (例えば、`mq.m5.large` クラスターには **`15000`**) を入力します。

    d. [Set / update limit] (制限を設定/更新) をクリックします。

    e. 上記のステップを繰り返します。[Limit] (制限) には、ドロップダウンオプションから [max-queues] を選択します。

14. 新しい制限が [仮想ホスト制限] リストに表示されていることを確認します。

RabbitMQ Management API を使用してデフォルトのポリシーと仮想ホスト制限を適用する

1. Amazon MQ コンソールにサインインします。
2. 左側のナビゲーションペインで [Brokers] (ブローカー) をクリックします。
3. ブローカーのリストから、新しいポリシーを適用するブローカーの名前を選択します。

4. ブローカーのページの [Connections] (接続) セクションで、RabbitMQ ウェブコンソール URL をメモします。これは、HTTP リクエストで使用するブローカーエンドポイントです。
5. 任意の新しいターミナルまたはコマンドラインウィンドウを開きます。
6. 新しいブローカーポリシーを作成するには、以下の curl コマンドを入力します。このコマンドでは、%2F としてエンコードされているデフォルト / vhost 上のキューを前提としています。別の vhost にポリシーを適用するには、%2F をその vhost の名前に置き換えてください。

> **Note**
>
> *#####*と*#####*を、管理者のサインイン認証情報に置き換えます。*number-of-messages* を、ブローカーのインスタンスサイズとデプロイモードに従った Amazon MQ の推奨値に置き換えます。*policy-name* をポリシーの名前に置き換えます。*broker-endpoint* を先ほどメモした URL に置き換えます。

```
curl -i -u username:password -H "content-type:application/json" -XPUT \
-d '{"pattern":".*", "priority":1, "definition":{"queue-mode":lazy,
 "overflow":"reject-publish", "max-length":"number-of-messages"}}' \
broker-endpoint/api/policies/%2F/policy-name
```

7. 新しいポリシーがブローカーのユーザーポリシーに追加されていることを確認するには、以下の curl コマンドを入力して、すべてのブローカーポリシーをリストします。

```
curl -i -u username:password broker-endpoint/api/policies
```

8. 新しい max-connections 仮想ホスト制限を作成するには、以下の curl コマンドを入力します。このコマンドでは、%2F としてエンコードされているデフォルト / vhost 上のキューを前提としています。別の vhost にポリシーを適用するには、%2F をその vhost の名前に置き換えてください。

> **Note**
>
> *#####*と*#####*を、管理者のサインイン認証情報に置き換えます。*max-connections* を、ブローカーのインスタンスサイズとデプロイモードに従った Amazon MQ の推奨値に置き換えます。ブローカーエンドポイントを先ほどメモした URL に置き換えます。

```
curl -i -u username:password -H "content-type:application/json" -XPUT \
-d '{"value":"number-of-connections"}' \
broker-endpoint/api/vhost-limits/%2F/max-connections
```

9. 新しい max-queues 仮想ホスト制限を作成するには、前のステップを繰り返しますが、curl コマンドを以下のように変更します。

```
curl -i -u username:password -H "content-type:application/json" -XPUT \
-d '{"value":"number-of-queues"}' \
broker-endpoint/api/vhost-limits/%2F/max-queues
```

10. 新しい制限がブローカーの仮想ホスト制限に追加されていることを確認するには、以下の curl コマンドを入力して、すべてのブローカー仮想ホスト制限をリストします。

```
curl -i -u username:password broker-endpoint/api/vhost-limits
```

Amazon MQ での RabbitMQ のクォーラムキュー

クォーラムキューは、1 つのリーダー (プライマリレプリカ) と複数のフォロワー (その他のレプリカ) で構成されるレプリケーション型のキュータイプです。リーダーが利用不可能になった場合、クォーラムキューでは、Raft コンセンサスアルゴリズムを使用して多数決で新しいリーダーノードが選出され、前のリーダーは同じクラスター内のフォロワーノードに降格されます。残りのフォロワーは、以前と同様にレプリケーションを続行します。各ノードは別々のアベイラビリティーゾーンにあるため、1 つのノードが一時的に利用できなくなっても、メッセージ配信は別のアベイラビリティーゾーンにある新しく選出されたリーダーレプリカによって続行されます。

クォーラムキューは、メッセージが失敗し、キューに何度も入れ直されたときに発生する有害メッセージを処理するために役立ちます。

以下の場合はクォーラムキューを使用しないでください。

- 一時キューを使用する場合
- 長いキューバックログがある場合
- 低レイテンシーを優先する場合

クォーラムキューを宣言するには、ヘッダー `x-queue-type` を `quorum` に設定します。

トピック

- Amazon MQ for RabbitMQ での従来のキューからクォーラムキューへの移行
- Amazon MQ for RabbitMQ のクォーラムキューのポリシー設定
- Amazon MQ for RabbitMQ のクォーラムキューのベストプラクティス

Amazon MQ for RabbitMQ での従来のキューからクォーラムキューへの移行

従来のミラーキューを、バージョン 3.13 以降の Amazon MQ ブローカーのクォーラムキューに移行することができます。そのためには、同じクラスター上に新しい仮想ホストを作成する方法と、インプレースで移行する方法があります。

オプション 1: 新しい仮想ホストを使用した従来のミラーキューからクォーラムキューへの移行

同じクラスター上に新しい仮想ホストを作成することで、従来のミラーキューをバージョン 3.13 以降の Amazon MQ ブローカーのクォーラムキューに移行できます。

1. 既存のクラスターで、デフォルトのキュータイプをクォーラムとする新しい仮想ホスト (vhost) を作成します。
2. 新しい vhost から フェデレーションプラグインを作成して、従来のミラーキューを使用する以前の vhost を指す URL を指定します。
3. `rabbitmqadmin` を使用して、以前の vhost から新しいファイルに定義を エクスポートします。このスキーマファイルを変更して、クォーラムキューとの互換性を持たせる必要があります。ファイルに加える必要のある変更の完全なリストについては、RabbitMQ クォーラムキュードキュメントの「Moving definitions」を参照してください。必要な変更をファイルに適用したら、新しい vhost に定義を再インポートします。
4. 新しい vhost に新しいポリシーを作成します。クォーラムキュー向けの Amazon MQ ポリシー設定に関する推奨事項については、「Amazon MQ for RabbitMQ のクォーラムキューのポリシー設定」を参照してください。次に、前の手順で作成した、以前の vhost から新しい vhost へのフェデレーションを開始します。
5. コンシューマーとプロデューサーが新しい vhost を指すように設定します。
6. Shovel プラグインを設定して、残っているメッセージをすべて移動します。キューが空になったら、Shovel を削除します。

従来のミラーキューからクォーラムキューへのインプレース移行

従来のミラーキューを、バージョン 3.13 以降の Amazon MQ ブローカーのクォーラムキューにインプレースで移行できます。

1. コンシューマーとプロデューサーを停止します。
2. 新しい一時クォーラムキューを作成します。
3. Shovel プラグインを設定して、以前の従来のミラーキューから新しい一時クォーラムキューにすべてのメッセージを移動します。すべてのメッセージが一時クォーラムキューに移動されたら、Shovel を削除します。
4. 移行元の従来のミラーキューを削除します。次に、移行元の従来のミラーキューと同じ名前とバインディングでクォーラムキューを再作成します。
5. 新しい Shovel を作成して、一時クォーラムキューから新しいクォーラムキューにメッセージを移動します。

Amazon MQ for RabbitMQ のクォーラムキューのポリシー設定

Amazon MQ 上の RabbitMQ ブローカーのクォーラムキューに、特定のポリシー設定を追加できます。

クォーラムキューのポリシーを作成するときは、次の操作を実行する必要があります。

- ha で始まるポリシー属性をすべて削除します。`ha-mode`、`ha-params`、`ha-sync-mode`、`ha-sync-batch-size`、`ha-promote-on-shutdown`、`ha-promote-on-failure` などが該当します。
- `queue-mode` を削除します。
- オーバーフローが `reject-publish-dlx` に設定されている場合は変更します。

> **Important**
>
> Amazon MQ for RabbitMQ は、ポリシー内のすべての属性を適用するか、一切適用しないかのどちらかの動作になります。従来のミラーキューとクォーラムキューの両方に適用されるポリシーを作成することはできません。ポリシーをクォーラムキューにのみ適用する場合は、`--apply-to` を `quorum_queues` に設定する必要があります。従来のミラーキューと

クォーラムキューを使用している場合は、クォーラムキューポリシーに加えて、`--apply-to:classic_queues` を設定した別のポリシーを作成する必要があります。

AWS-DEFAULT ポリシーは、「適用先」パラメータに自動的に新しいキュータイプを採用するため、変更する必要はありません。Amazon MQ for RabbitMQ のデフォルトポリシーの詳細については、「オペレーターポリシーの設定」を参照してください。

Amazon MQ for RabbitMQ のクォーラムキューのベストプラクティス

クォーラムキューを使用する際のパフォーマンスを向上させるには、次のベストプラクティスを使用することをお勧めします。

配信制限を設定して有害メッセージを処理する

有害メッセージは、メッセージが失敗し、何度も再配信される場合に発生します。`delivery-limit` ポリシー引数を使用してメッセージ配信制限を設定すると、何度も再配信されるメッセージを削除できます。配信制限で許容される回数を超えてメッセージが再配信された場合、メッセージは RabbitMQ によってドロップされ、削除されます。配信制限を設定すると、メッセージはキューの先頭の近くに再び入れられます。

クォーラムキューのメッセージ優先度

クォーラムキューにはメッセージ優先度がありません。メッセージ優先度が必要な場合は、複数のクォーラムキューを作成する必要があります。複数のクォーラムキューを持つメッセージの優先度設定の詳細については、RabbitMQ ドキュメントの「Message priority」を参照してください。

デフォルトのレプリケーション係数の使用

Amazon MQ for RabbitMQ では、クォーラムキューを使用するクラスターブローカーのレプリケーション係数はデフォルトで 3 ノードになります。`x-quorum-initial-group-size` に変更を加えると、Amazon MQ は、再びデフォルトでレプリケーション係数 3 を使用するようになります。

Amazon MQ for RabbitMQ のベストプラクティス

Amazon MQ for RabbitMQ ブローカーを使用する際にブローカーのパフォーマンスを最大化し、メッセージのスループット効率を最適化するには、以下の本番の準備状況ガイドラインに従ってください。

> **Important**
>
> 現在、Amazon MQ はストリームや、RabbitMQ 3.9.x で導入された JSON での構造化ロギングの使用をサポートしていません。

トピック

- Amazon MQ for RabbitMQ でのブローカーのセットアップと接続管理のベストプラクティス
- Amazon MQ for RabbitMQ でのメッセージの耐久性と信頼性に関するベストプラクティス
- Amazon MQ for RabbitMQ のパフォーマンス最適化と効率のベストプラクティス
- Amazon MQ for RabbitMQ でのネットワークの耐障害性とモニタリングのベストプラクティス

Amazon MQ for RabbitMQ でのブローカーのセットアップと接続管理のベストプラクティス

ブローカーのセットアップと接続管理は、ブローカーメッセージのスループット、リソース使用率、本番環境のワークロードを処理する機能に関する問題を防ぐ最初のステップです。Amazon MQ for RabbitMQ ブローカーを作成および設定するときは、適切なインスタンスタイプの選択、接続の効率的な管理、ブローカーのパフォーマンスを最大化するためのメッセージプリフェッチの設定に関する以下のベストプラクティスを完了してください。

> **Important**
>
> Amazon MQ for RabbitMQ では、ユーザー名「guest」はサポートされず、デフォルトのゲストアカウントは新しいブローカーの作成時に削除されます。ユーザーが作成した「guest」というアカウントも、Amazon MQ によって定期的に削除されます。

ステップ 1: クラスターデプロイを使用する

本番ワークロードでは、高可用性とメッセージの耐障害性を確保するために、単一インスタンスブローカーの代わりにクラスターデプロイを使用することをお勧めします。クラスターデプロイでは、単一障害点が削除され、耐障害性が向上します。

クラスターデプロイは、3 つのアベイラビリティーゾーンに分散された 3 つの RabbitMQ ブローカーノードで構成され、自動フェイルオーバーを提供し、アベイラビリティーゾーン全体が使用できなく

なってもオペレーションが続行されるようにします。Amazon MQ は、すべてのノードにメッセージを自動的にレプリケートし、ノードの障害時やメンテナンス時の可用性を確保します。

クラスターのデプロイは本番環境に不可欠であり、Amazon MQ サービスレベルアグリーメントでサポートされています。

詳細については、「Amazon MQ for RabbitMQ でのクラスターデプロイ」を参照してください。

ステップ 2: 正しいブローカーインスタンスタイプを選択する

ブローカーインスタンスタイプのメッセージスループットは、アプリケーションのユースケースによって異なります。`M7g.medium` はアプリケーションパフォーマンスのテストにのみ使用する必要があります。本番環境で大きいインスタンスを使用する前に、この小さいインスタンスを使用すると、アプリケーションのパフォーマンスを向上させることができます。`m7g.large` 以上のインスタンスタイプでは、クラスターデプロイを使用して高可用性とメッセージの耐久性を確保できます。大きいブローカーインスタンスタイプは、本番稼働レベルのクライアントとキュー、高スループット、メモリ内のメッセージ、冗長メッセージを処理できます。

適切なインスタンスタイプを選択する方法の詳細については、「Amazon MQ for RabbitMQ のサイズ設定ガイドライン」を参照してください。

ステップ 3: クォーラムキューを使用する

クラスターデプロイのクォーラムキューは、3.13 以降の RabbitMQ ブローカーの本番環境でレプリケートされたキュータイプのデフォルトの選択肢である必要があります。クォーラムキューは、高い信頼性、高スループット、安定したレイテンシーを提供する最新のレプリケートキュータイプです。

クォーラムキューは Raft コンセンサスアルゴリズムを使用して耐障害性を向上させます。リーダーノードが使用できなくなると、クォーラムキューは過半数投票によって自動的に新しいリーダーを選択し、メッセージ配信が中断を最小限に抑えながら継続されるようにします。各ノードは異なるアベイラビリティーゾーンにあるため、1 つのアベイラビリティーゾーン全体が一時的に使用できなくなってもメッセージングシステムは引き続き使用できます。

クォーラムキューを宣言するには、キューの作成時に `x-queue-type` ヘッダーを `quorum` に設定します。

移行戦略やベストプラクティスなど、クォーラムキューの詳細については、「Amazon MQ for RabbitMQ のクォーラムキュー」を参照してください。

ステップ 4: 複数のチャネルを使用する

接続チャーンを回避するには、1 つの接続で複数のチャネルを使用します。アプリケーションでは、チャネルに対する 1:1 の接続を避ける必要があります。プロセスごとに 1 つの接続を使用し、スレッドごとに 1 つのチャネルを使用することをお勧めします。チャネルのリークを防ぐために、チャネルを過剰に使用することは避けてください。

Amazon MQ for RabbitMQ でのメッセージの耐久性と信頼性に関するベストプラクティス

アプリケーションを本番環境に移行する前に、メッセージの損失とリソースの過剰使用を防ぐための以下のベストプラクティスを完了してください。

ステップ 1: 永続メッセージと持続キューを使用する

永続メッセージは、ブローカーがクラッシュまたは再起動するという状況におけるデータの耐久性の保護に役立ちます。永続メッセージは、到着するとすぐにディスクに書き込まれますが、レイジーキューとは異なり、ブローカーがより多くのメモリを必要とする場合を除き、永続メッセージはメモリとディスクの両方にキャッシュされます。より多くのメモリが必要な場合は、ディスクへのメッセージの保存を管理する RabbitMQ ブローカーメカニズム (一般に永続レイヤーと呼ばれます) によって、メモリからメッセージが削除されます。

メッセージの永続性を有効にするには、キューを `durable` として宣言し、メッセージ配信モードを `persistent` に設定できます。以下の例は、RabbitMQ Java クライアントライブラリを使用した持続キューの宣言を示しています。AMQP 0-9-1 を使用している場合は、配信モード「2」を設定することで、メッセージを永続としてマークできます。

```
boolean durable = true;
channel.queueDeclare("my_queue", durable, false, false, null);
```

キューを持続キューとして設定したら、以下の例にあるように、`MessageProperties` を `PERSISTENT_TEXT_PLAIN` に設定することによって永続メッセージをキューに送信できます。

```
import com.rabbitmq.client.MessageProperties;

channel.basicPublish("", "my_queue",
            MessageProperties.PERSISTENT_TEXT_PLAIN,
            message.getBytes());
```

ステップ 2: パブリッシャーの確認とコンシューマーの配信承認の設定

ブローカーにメッセージが送信されたことを確認するプロセスは、パブリッシャーの確認と呼ばれます。パブリッシャーは、メッセージが確実に格納されたときにアプリケーションに通知します。パブリッシャーの確認は、ブローカーに格納されるメッセージの割合を制御するためにも役立ちます。パブリッシャーの確認がないと、メッセージが正常に処理されたという確認が得られないため、ブローカーは処理できないメッセージをドロップする可能性があります。

同様に、クライアントアプリケーションはメッセージの配信と消費の確認をブローカーに返送します。これはコンシューマーの配信承認と呼ばれます。RabbitMQ ブローカーの使用時にデータの安全性を確保するには、確認と承認の両方が不可欠です。

コンシューマーの配信承認は、通常クライアントアプリケーションで設定されています。AMQP 0-9-1 を使用している場合は、`basic.consume` メソッドを設定することで承認を有効化できます。AMQP 0-9-1 クライアントでは、`confirm.select` メソッドを送信してパブリッシャーの確認を設定することもできます。

通常、配信承認はチャネルで有効化されます。例えば、RabbitMQ Java クライアントライブラリの使用時には、以下の例にあるように、`Channel#basicAck` を使用してシンプルな `basic.ack` 肯定承認をセットアップできます。

```
// this example assumes an existing channel instance

boolean autoAck = false;
channel.basicConsume(queueName, autoAck, "a-consumer-tag",
     new DefaultConsumer(channel) {
         @Override
         public void handleDelivery(String consumerTag,
                                    Envelope envelope,
                                    AMQP.BasicProperties properties,
                                    byte[] body)
             throws IOException
         {
             long deliveryTag = envelope.getDeliveryTag();
             // positively acknowledge a single delivery, the message will
             // be discarded
             channel.basicAck(deliveryTag, false);
         }
     });
```

> **Note**
>
> 未承認メッセージは、メモリにキャッシュする必要があります。コンシューマーがプリフェッチするメッセージの数は、クライアントアプリケーションのプリフェッチを設定することによって制限できます。

consumer_timeout を設定すると、コンシューマーから配信承認が届かない状況を検出できます。コンシューマーがタイムアウト値の時間内に承認を送信しない場合、チャネルは閉じられ、PRECONDITION_FAILED が発生します。エラーを診断するには、UpdateConfiguration API を使用して consumer_timeout 値を大きくします。

ステップ 3: キューを短くしておく

クラスターデプロイでは、多数のメッセージを持つキューがリソースの過剰な使用につながる場合があります。ブローカーが過剰に使用されているときは、Amazon MQ for RabbitMQ ブローカーの再起動がパフォーマンスをさらに低下させる原因となる可能性があります。過剰に使用されているブローカーが再起動されると、REBOOT_IN_PROGRESS 状態のまま応答しなくなることがあります。

Amazon MQ はメンテナンスウィンドウ中、すべてのメンテナンス作業を一度に 1 ノードずつ実行して、ブローカーが動作可能な状態を維持することを確実にします。その結果、各ノードが操作を再開するときに、キューが同期する必要が生じる場合があります。同期中、ミラーにレプリケートする必要があるメッセージは、バッチで処理されるように、対応する Amazon Elastic Block Store (Amazon EBS) ボリュームからメモリにロードされます。メッセージをバッチで処理することにより、キューの同期が速くなります。

キューを短くし、メッセージを小さくしておくと、キューが正常に同期し、期待通りに操作を再開します。ただし、バッチ内のデータ量がノードのメモリ制限に近づいた場合は、ノードが高メモリアラームを発し、キューの同期を一時停止します。メモリ使用量は、CloudWatch で RabbitMemUsed および RabbitMqMemLimit のブローカーノードメトリクスを比較することで確認できます。同期は、メッセージが消費もしくは削除される、またはバッチ内のメッセージの数が減るまで完了できません。

クラスターデプロイのためにキューの同期化が一時停止される場合は、メッセージを消費または削除して、キュー内のメッセージの数を減らすことをお勧めします。キュー深度が減少し、キューの同期が完了すると、ブローカーのステータスが RUNNING に変更されます。一時停止されたキューの同期を解決するには、キューの同期のバッチサイズを小さくするポリシーを適用することも可能です。

また、自動削除ポリシーと TTL ポリシーを定義すると、リソースの使用量をプロアクティブに削減するとともに、コンシューマーからの NACK を最小限に抑えることができます。ブローカーへのメッセージの再キュー処理は CPU 負荷が高いため、大量の NACK が発生するとブローカーのパフォーマンスに影響する可能性があります。

Amazon MQ for RabbitMQ のパフォーマンス最適化と効率のベストプラクティス

Amazon MQ for RabbitMQ ブローカーのパフォーマンスを最適化するには、スループットを最大化し、レイテンシーを最小限に抑え、効率的なリソース使用率を確保します。アプリケーションのパフォーマンスを最適化するには、次のベストプラクティスを実行します。

ステップ 1: メッセージサイズを 1 MB 未満に維持する

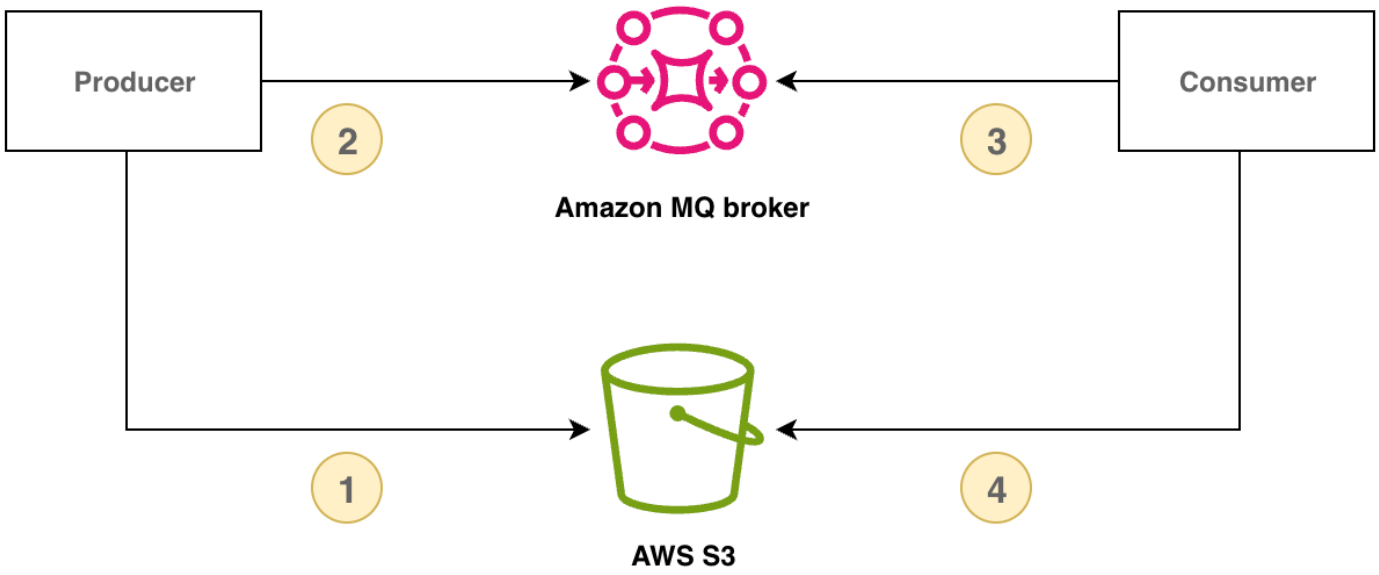
最適なパフォーマンスと信頼性を得るには、メッセージを 1 メガバイト (1 MB) 未満にしておくことをお勧めします。

RabbitMQ 3.13 はデフォルトで最大 128 MB のメッセージサイズをサポートしますが、大きなメッセージは、発行をブロックし、ノード間でメッセージをレプリケートしながら高いメモリ負荷を発生させる可能性のある予測不可能なメモリアラームをトリガーする可能性があります。メッセージのサイズが大きすぎると、ブローカーの再起動プロセスや復旧プロセスにも影響し、サービス継続性に対するリスクが高まり、パフォーマンスが低下する可能性があります。

クレームチェックパターンを使用して大きなペイロードを保存および取得する

大きなメッセージを管理するには、メッセージペイロードを外部ストレージに保存し、RabbitMQ を介してペイロード参照識別子のみを送信することにより、クレームチェックパターンを実装できます。コンシューマーはペイロード参照識別子を使用して、大きなメッセージを取得して処理します。

次の図は、Amazon MQ for RabbitMQ と Amazon S3 を使用してクレームチェックパターンを実装する方法を示しています。



次の例は、Amazon MQ、AWS SDK for Java 2.x、Amazon S3 を使用したこのパターンを示しています。

1. まず、Amazon S3 参照識別子を保持する Message クラスを定義します。

```
class Message {
    // Other data fields of the message...

    public String s3Key;
    public String s3Bucket;
}
```

2. Amazon S3 にペイロードを保存し、RabbitMQ を介してリファレンスメッセージを送信するパブリッシャーメソッドを作成します。

```
public void publishPayload() {
    // Store the payload in S3.
    String payload = PAYLOAD;
    String prefix = S3_KEY_PREFIX;
    String s3Key = prefix + "/" + UUID.randomUUID();
    s3Client.putObject(PutObjectRequest.builder()
        .bucket(S3_BUCKET).key(s3Key).build(),
        RequestBody.fromString(payload));

    // Send the reference through RabbitMQ.
    Message message = new Message();
```

```
    message.s3Key = s3Key;
    message.s3Bucket = S3_BUCKET;
    // Assign values to other fields in your message instance.

    publishMessage(message);
}
```

3. Amazon S3 からペイロードを取得し、ペイロードを処理して、Amazon S3 オブジェクトを削除するコンシューマーメソッドを実装します。

```
public void consumeMessage(Message message) {
    // Retrieve the payload from S3.
    String payload = s3Client.getObjectAsBytes(GetObjectRequest.builder()
        .bucket(message.s3Bucket).key(message.s3Key).build())
        .asUtf8String();

    // Process the complete message.
    processPayload(message, payload);

    // Delete the S3 object.
    s3Client.deleteObject(DeleteObjectRequest.builder()
        .bucket(message.s3Bucket).key(message.s3Key).build());
}
```

ステップ 2: **basic.consume** と存続期間の長いコンシューマーを使用する

存続期間の長いコンシューマーで basic.consume を使用すると、basic.get を使用して個々のメッセージをポーリングするよりも効率的です。詳細については、「個々のメッセージのポーリング」を参照してください。

ステップ 3: プリフェッチを設定する

RabbitMQ のプリフェッチ値を使用して、コンシューマーがメッセージを消費する方法を最適化できます。RabbitMQ は、プリフェッチ数をチャネルではなくコンシューマーに適用することによって、AMQP 0-9-1 が提供するチャネルプリフェッチメカニズムを実装します。プリフェッチ値は、特定の時間にコンシューマに送信されるメッセージの数を指定するために使用されます。デフォルトで、RabbitMQ はクライアントアプリケーションに無制限のバッファサイズを設定します。

RabbitMQ コンシューマーにプリフェッチ数を設定するときに考慮する要因にはさまざまなものがあります。まず、コンシューマーの環境と設定を考慮します。コンシューマーは、メッセージが処理されるときにそれらすべてをメモリに保持する必要があるため、高いプリフェッチ値はコンシューマー

のパフォーマンスに悪影響を及ぼし、場合によってはコンシューマー全体がクラッシュする原因になることもあります。同様に、RabbitMQ ブローカー自体も、コンシューマー承認を受け取るまで、送信するすべてのメッセージをメモリにキャッシュしておきます。コンシューマに自動承認が設定されておらず、コンシューマによるメッセージの処理に比較的長い時間がかかる場合、高いプリフェッチ値は RabbitMQ サーバーのメモリがすぐになくなる原因になる可能性があります。

上記の考慮事項を踏まえて、大量の未処理または未承認のメッセージが原因で RabbitMQ ブローカー、またはそのコンシューマーでメモリ不足が発生する状況を防ぐため、常にプリフェッチ値を設定することが推奨されます。大量のメッセージを処理するためにブローカーを最適化する必要がある場合は、さまざまなプリフェッチ数を使用してブローカーとコンシューマーをテストし、コンシューマーがメッセージを処理するためにかかる時間と比較して、ネットワークオーバーヘッドがおおむね軽微なものになる値を判断します。

> **Note**
>
> - コンシューマーへのメッセージの配信を自動承認するようにクライアントアプリケーションが設定されている場合、プリフェッチ値を設定しても効果はありません。
> - プリフェッチされたメッセージはすべて、キューから削除されます。

以下の例は、RabbitMQ Java クライアントライブラリを使用した単一のコンシューマーへのプリフェッチ値 `10` の設定を示しています。

```
ConnectionFactory factory = new ConnectionFactory();

Connection connection = factory.newConnection();
Channel channel = connection.createChannel();

channel.basicQos(10, false);

QueueingConsumer consumer = new QueueingConsumer(channel);
channel.basicConsume("my_queue", false, consumer);
```

> **Note**
>
> RabbitMQ Java クライアントライブラリでは、`global` フラグのデフォルト値が `false` に設定されているので、上記の例は単純に `channel.basicQos(10)` として記述できます。

ステップ 4: Celery 5.5 以降をクォーラムキューで使用する

分散タスクキューシステムである Python Celery は、高いタスクロードが発生している場合に、重要ではないメッセージを多数生成できます。この追加のブローカーアクティビティにより the section called “ RABBITMQ_MEMORY_ALARM ” がトリガーされ、ブローカーが利用できなくなる可能性があります。メモリアラームがトリガーされる可能性を減らすには、以下を実行します。

すべての Celery バージョンの場合

1. `task_create_missing_queues` をオフにしてキューチャーンを軽減します。
2. 次に、`worker_enable_remote_control` をオフにして、`celery@...pidbox` キューの動的作成を停止します。これにより、ブローカーのキューチャーンが減少します。

```
worker_enable_remote_control = false
```

3. 重要でないメッセージアクティビティをさらに減らすには、Celery アプリケーションを起動するときに `-E` または `--task-events` フラグを付けずに、Celery worker-send-task-events をオフにします。
4. 次のパラメータを使用して Celery アプリケーションを起動します。

```
celery -A app_name worker --without-heartbeat --without-gossip --without-mingle
```

Celery バージョン 5.5 以降の場合

1. Celery バージョン 5.5、クォーラムキューをサポートする最小バージョン、またはそれ以降のバージョンにアップグレードします。使用している Celery のバージョンを確認するには、`celery --version` を使用します。クォーラムキューの詳細については、「the section called “クォーラムキュー”」を参照してください。
2. Celery 5.5 以降にアップグレードした後、`task_default_queue_type` を "quorum" に設定します。
3. 次に、ブローカートランスポートオプションで発行確認を有効にする必要もあります。

```
broker_transport_options = {"confirm_publish": True}
```

Amazon MQ for RabbitMQ でのネットワークの耐障害性とモニタリングのベストプラクティス

信頼性の高いメッセージングアプリケーションを維持するには、ネットワークの耐障害性とブローカーメトリクスのモニタリングが不可欠です。自動復旧メカニズムとリソースモニタリング戦略を実装するには、次のベストプラクティスを実行します。

ステップ 1: ネットワーク障害から自動的に回復する

RabbitMQ ノードへのクライアント接続が失敗した場合の大幅なダウンタイムを防ぐため、自動ネットワークリカバリを常に有効にしておくことをお勧めします。バージョン `4.0.0` 以降の RabbitMQ Java クライアントライブラリは、自動ネットワークリカバリをデフォルトでサポートします。

自動接続リカバリは、接続の I/O ループで未処理の例外がスローされた場合、ソケット読み取り操作のタイムアウトが検出された場合、またはサーバーがハートビートを受信しない場合にトリガーされます。

クライアントと RabbitMQ ノード間の初期接続が失敗した場合、自動リカバリはトリガーされません。アプリケーションコードは、接続の再試行によって、初期接続障害を考慮するように記述することをお勧めします。以下の例は、RabbitMQ Java クライアントライブラリを使用した初期ネットワーク障害の再試行を示しています。

```
ConnectionFactory factory = new ConnectionFactory();
// enable automatic recovery if using RabbitMQ Java client library prior to version
 4.0.0.
factory.setAutomaticRecoveryEnabled(true);
// configure various connection settings

try {
  Connection conn = factory.newConnection();
} catch (java.net.ConnectException e) {
  Thread.sleep(5000);
  // apply retry logic
}
```

> **Note**
>
> アプリケーションが `Connection.Close` メソッド使用して接続を閉じる場合、自動ネットワークリカバリは有効化またはトリガーされません。

ステップ 2: ブローカーメトリクスとアラームをモニタリングする

Amazon MQ for RabbitMQ ブローカーの CloudWatch メトリクスとアラームを定期的にモニタリングして、メッセージングアプリケーションに影響を与える前に潜在的な問題を特定して対処することをお勧めします。プロアクティブモニタリングは、回復力のあるメッセージングアプリケーションを維持し、最適なパフォーマンスを確保するために不可欠です。

Amazon MQ for RabbitMQ は、ブローカーのパフォーマンス、リソース使用率、メッセージフローに関するインサイトを提供するメトリクスを CloudWatch に発行します。モニタリングする主なメトリクスには、メモリ使用量とディスク使用量が含まれます。ブローカーがリソース制限に近づいたり、パフォーマンスが低下したりしたときの CloudWatch アラームを設定できます。

次の必須メトリクスをモニタリングします。

`RabbitMQMemUsed` および `RabbitMQMemLimit`

メモリ使用量をモニタリングして、メッセージの発行をブロックする可能性のあるメモリアラームを防止します。

`RabbitMQDiskFree` および `RabbitMQDiskFreeLimit`

ディスクの使用状況をモニタリングして、ブローカーの障害の原因となるディスク容量の問題を回避します。

クラスターデプロイでは、ノード固有のメトリクスもモニタリングして、ノード固有の問題を特定します。

> **Note**
>
> 高メモリアラームを防ぐ方法の詳細については、「Address and prevent high memory alarm」を参照してください。

RabbitMQ のチュートリアル

以下のチュートリアルでは、Amazon MQ で RabbitMQ を設定して使用する方法を説明します。サポートされているクライアントライブラリを Node.js、Python、.NET などのさまざまなプログラミング言語で使用する方法の詳細については、「RabbitMQ Getting Started Guide」の「RabbitMQ Tutorials」を参照してください。

トピック

ブローカー設定の編集

を使用して CloudWatch ログの有効化や無効化など、ブローカーの設定を編集できます AWS マネジメントコンソール。

RabbitMQ ブローカーオプションを編集する

1. Amazon MQ コンソールにサインインします。
2. ブローカーリストからブローカーを選択して (MyBroker など)、[Edit] (編集) をクリックします。
3. [***MyBroker*** の編集] ページの [仕様] セクションで、[ブローカーエンジンのバージョン] または [ブローカーインスタンスタイプ] を選択します。
4. [CloudWatch Logs] セクションのトグルボタンをクリックして、一般ログを有効化または無効化します。これ以上のステップは必要ありません。

> **Note**
>
> - RabbitMQ ブローカーの場合、Amazon MQ は自動的にサービスリンクロール (SLR) を使用して、CloudWatch に一般ログを発行します。詳細については、「the section called “サービスリンクロールの使用”」を参照してください。
> - Amazon MQ は、RabbitMQ ブローカーに対する監査ロギングをサポートしません。

5. [Maintenance (メンテナンス)] セクションで、ブローカーのメンテナンススケジュールを設定します。

   ブローカーを AWS リリース時に新しいバージョンにアップグレードするには、「自動マイナーバージョンアップグレードを有効にする」を選択します。自動アップグレードは、曜日、時刻 (24 時間形式)、およびタイムゾーン (デフォルトは UTC) で定義されたメンテナンスウィンドウ中に行われます。

6. [Schedule modifications (スケジュールの変更)] を選択します。

   > **Note**
   >
   > [自動マイナーバージョンのアップグレードを有効にする] のみを選択した場合、ブローカーの再起動が必要ないため、ボタンは [保存] に変わります。

   設定が指定された時刻にブローカーに適用されます。

Amazon MQ for RabbitMQ でPython Pika を使う

次のチュートリアルでは、Amazon MQ for RabbitMQ ブローカーに接続するように構成された TLS を使用して Python Pika クライアントをセットアップする方法を示しています。Pika は RabbitMQ のための AMQP 0-9-1 プロトコルの Python 実装です。このチュートリアルでは、Pika のインストール、キューの宣言、ブローカーのデフォルトエクスチェンジにメッセージを送信するパブリッシャーの設定、キューからメッセージを受信するコンシューマーの設定について説明します。

トピック

- 前提条件
- 権限

前提条件

このチュートリアルの最初のステップを完了するには、以下のものが必要です。

- Amazon MQ for RabbitMQ ブローカー。詳細については、「Amazon MQ for RabbitMQ ブローカーを作成する」を参照してください。
- オペレーティングシステム用に Python 3 がインストールされています。
- Python `pip` を使用して、Pika がインストールされました。Pika をインストールするには、新しいターミナルウィンドウを開き、以下を実行します。

```
$ python3 -m pip install pika
```

権限

このチュートリアルでは、vhost への書き込みおよび読み取りの許可を持つ Amazon MQ for RabbitMQ ブローカーユーザーが少なくとも 1 人必要です。次の表は、正規表現 (regexp) パターンに必要な最小アクセス許可を示しています。

タグ	設定 regexp	書き込み regexp	読み込み regexp
none		`.*`	`.*`

リストされているユーザー許可は、ブローカで管理オペレーションを実行するための管理プラグインへのアクセスを付与することなく、ユーザーに読み取りおよび書き込み許可のみを提供します。特定のキューへのユーザーのアクセスを制限する正規表現パターンを提供することで、許可をさらに制限できます。例えば、読み取り regexp パターンを `^[hello world].*` に変更する場合、ユーザーには `hello world` で始まるキューからの読み取り許可のみが付与されます。

RabbitMQ ユーザーの作成、およびユーザータグと許可の管理の詳細については、「Amazon MQ for RabbitMQ ブローカーのユーザー」を参照してください。

ステップ 1: 基本的な Python Pika クライアントを作成する

Amazon MQ for RabbitMQ ブローカーと対話するときに、コンストラクタを定義し、TLS 設定に必要な SSL コンテキストを提供する Python Pika クライアント基本クラスを作成するには、次の手順を実行します。

1. 新しいターミナルウィンドウを開き、プロジェクトの新しいディレクトリを作成し、そのディレクトリに移動します。

```
$ mkdir pika-tutorial
$ cd pika-tutorial
```

2. 以下の Python コードを含む basicClient.py というファイルを作成します。

```
import ssl
import pika

class BasicPikaClient:

    def __init__(self, rabbitmq_broker_id, rabbitmq_user, rabbitmq_password,
 region):

        # SSL Context for TLS configuration of Amazon MQ for RabbitMQ
        ssl_context = ssl.SSLContext(ssl.PROTOCOL_TLSv1_2)
        ssl_context.set_ciphers('ECDHE+AESGCM:!ECDSA')

        url = f"amqps://{rabbitmq_user}:
{rabbitmq_password}@{rabbitmq_broker_id}.mq.{region}.amazonaws.com:5671"
        parameters = pika.URLParameters(url)
        parameters.ssl_options = pika.SSLOptions(context=ssl_context)

        self.connection = pika.BlockingConnection(parameters)
        self.channel = self.connection.channel()
```

パブリッシャーとコンシューマに対して、BasicPikaClient から継承する追加のクラスを定義できるようになりました。

ステップ 2: パブリッシャーを作成してメッセージを送信する

キューを宣言し、1 つのメッセージを送信するパブリッシャを作成するには、次の手順を実行します。

1. 次のコードサンプルの内容をコピーし、前のステップで作成した同じディレクトリで、publisher.py と名前を付けてローカルに保存します。

```
from basicClient import BasicPikaClient

class BasicMessageSender(BasicPikaClient):

    def declare_queue(self, queue_name):
        print(f"Trying to declare queue({queue_name})...")
        self.channel.queue_declare(queue=queue_name)

    def send_message(self, exchange, routing_key, body):
        channel = self.connection.channel()
        channel.basic_publish(exchange=exchange,
                              routing_key=routing_key,
                              body=body)
        print(f"Sent message. Exchange: {exchange}, Routing Key: {routing_key},
 Body: {body}")

    def close(self):
        self.channel.close()
        self.connection.close()

if __name__ == "__main__":

    # Initialize Basic Message Sender which creates a connection
    # and channel for sending messages.
    basic_message_sender = BasicMessageSender(
        "<broker-id>",
        "<username>",
        "<password>",
        "<region>"
    )

    # Declare a queue
    basic_message_sender.declare_queue("hello world queue")

    # Send a message to the queue.
```

```
    basic_message_sender.send_message(exchange="", routing_key="hello world queue",
body=b'Hello World!')

    # Close connections.
    basic_message_sender.close()
```

BasicMessageSender クラスは BasicPikaClient から継承され、キューの宣言、キューへのメッセージの送信、および接続を閉じるための追加のメソッドを実装します。コードサンプルでは、キューの名前と等しいルーティングキーを使用して、メッセージをデフォルトの交換にルーティングします。

2. [if __name__ == "__main__":] で、渡されたパラメータを次の情報を含む BasicMessageSender コンストラクターステートメントで置換します。

   - **<broker-id>** – Amazon MQ がブローカー用に生成する一意の ID です。ID は、ブローカー ARN から解析できます。例えば、arn:aws:mq:us-east-2:123456789012:broker:MyBroker:b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9 という ARN の場合、ブローカー ID は b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9 になります。
   - **<username>** - ブローカにメッセージを書き込むのに十分な許可を持つブローカユーザーのユーザー名。
   - **<password>** - ブローカにメッセージを書き込むのに十分な許可を持つブローカユーザーのパスワード。
   - **<region>** – Amazon MQ for RabbitMQ ブローカーを作成した AWS リージョン。例えば、us-west-2。

3. publisher.py を作成した同じディレクトリで次のコマンドを実行します。

```
$ python3 publisher.py
```

コードが正常に実行された場合、ターミナルウィンドウに次の出力が表示されます。

```
Trying to declare queue(hello world queue)...
Sent message. Exchange: , Routing Key: hello world queue, Body: b'Hello World!'
```

ステップ 3: コンシューマーを作成してメッセージを受信する

キューから単一のメッセージを受信するコンシューマーを作成するには、次の手順を実行します。

1. 次のコードサンプルの内容をコピーし、同じディレクトリで、consumer.py と名前を付けてローカルに保存します。

```
from basicClient import BasicPikaClient

class BasicMessageReceiver(BasicPikaClient):

    def get_message(self, queue):
        method_frame, header_frame, body = self.channel.basic_get(queue)
        if method_frame:
            print(method_frame, header_frame, body)
            self.channel.basic_ack(method_frame.delivery_tag)
            return method_frame, header_frame, body
        else:
            print('No message returned')

    def close(self):
        self.channel.close()
        self.connection.close()


if __name__ == "__main__":

    # Create Basic Message Receiver which creates a connection
    # and channel for consuming messages.
    basic_message_receiver = BasicMessageReceiver(
        "<broker-id>",
        "<username>",
        "<password>",
        "<region>"
    )

    # Consume the message that was sent.
    basic_message_receiver.get_message("hello world queue")

    # Close connections.
    basic_message_receiver.close()
```

前のステップで作成したパブリッシャーと同様に、 は からBasicMessageReceiver継承BasicPikaClientし、単一のメッセージを受信して接続を閉じるための追加のメソッドを実装します。

2. `if __name__ == "__main__":` ステートメントで、渡されたパラメータを次の情報を含む `BasicMessageReceiver` コンストラクターに置換します。
3. プロジェクトディレクトリで次のコマンドを実行します。

```
$ python3 consumer.py
```

コードが正常に実行されると、メッセージ本文とルーティングキーを含むヘッダーがターミナルウィンドウに表示されます。

```
<Basic.GetOk(['delivery_tag=1', 'exchange=', 'message_count=0',
 'redelivered=False', 'routing_key=hello world queue'])> <BasicProperties> b'Hello
 World!'
```

ステップ 4: (オプション) イベントループを設定し、メッセージを消費する

キューから複数のメッセージを消費するには、Pika の `basic_consume` メソッドと、次に示すコールバック関数を使用します

1. `consumer.py` で、`BasicMessageReceiver` クラスに以下のメソッド定義を追加します。

```
def consume_messages(self, queue):
    def callback(ch, method, properties, body):
        print(" [x] Received %r" % body)

    self.channel.basic_consume(queue=queue, on_message_callback=callback,
 auto_ack=True)

    print(' [*] Waiting for messages. To exit press CTRL+C')
    self.channel.start_consuming()
```

2. `consumer.py` の `if __name__ == "__main__":` の下で、前のステップで定義した `consume_messages` メソッドを呼び出します。

```
if __name__ == "__main__":

    # Create Basic Message Receiver which creates a connection and channel for
 consuming messages.
    basic_message_receiver = BasicMessageReceiver(
        "<broker-id>",
```

```
        "<username>",
        "<password>",
        "<region>"
    )

    # Consume the message that was sent.
    # basic_message_receiver.get_message("hello world queue")

    # Consume multiple messages in an event loop.
    basic_message_receiver.consume_messages("hello world queue")

    # Close connections.
    basic_message_receiver.close()
```

3. consumer.py をもう一度実行し、成功すると、キューに入れられたメッセージがターミナルウィンドウに表示されます。

```
[*] Waiting for messages. To exit press CTRL+C
[x] Received b'Hello World!'
[x] Received b'Hello World!'
...
```

次のステップ

- サポートされている他の RabbitMQ クライアントライブラリの詳細については、RabbitMQ のウェブサイトの「RabbitMQ クライアントドキュメント」を参照してください。

RabbitMQ の一時停止されたキュー同期の解決

Amazon MQ for RabbitMQ クラスターデプロイでは、各キューに発行されたメッセージが 3 つのブローカーノード全体にレプリケートされます。ミラーリングと呼ばれるこのレプリケーションは、RabbitMQ ブローカーに高可用性 (HA) を提供します。クラスターデプロイ内のキューは、1 つのノード上にあるメインレプリカと、1 つ、または複数のミラーで構成されています。ミラーキューに適用されるすべての操作 (メッセージのキュー登録など) は、まずメインキューに適用され、その後ミラー全体にレプリケートされます。

例えば、メインノード (main) と 2 つのミラー (mirror-1 および mirror-2) の 3 つのノード全体にレプリケートされたミラーキューについて考えてみましょう。このミラーキュー内のすべてのメッセージがすべてのミラーに正常に伝播されると、キューが同期されたことになります。ノード

(mirror-1) が一定期間使用できなくなった場合でも、キューは引き続き動作可能で、メッセージのキュー登録を継続できますが、キューを同期するには、mirror-1 が使用不可である間に main に発行されたメッセージが mirror-1 にレプリケートされる必要があります。

ミラーリングの詳細については、RabbitMQ ウェブサイトで「Classic Mirrored Queues」を参照してください。

メンテナンスとキューの同期

メンテナンスウィンドウ中、Amazon MQ はすべてのメンテナンス作業を一度に 1 ノードずつ実行して、ブローカーが動作可能な状態を維持することを確実にします。その結果、各ノードが操作を再開するときに、キューが同期する必要が生じる場合があります。同期中、ミラーにレプリケートする必要があるメッセージは、バッチで処理されるように、対応する Amazon Elastic Block Store (Amazon EBS) ボリュームからメモリにロードされます。メッセージをバッチで処理することにより、キューの同期が速くなります。

キューを短くし、メッセージを小さくしておくと、キューが正常に同期し、期待通りに操作を再開します。ただし、バッチ内のデータ量がノードのメモリ制限に近づいた場合は、ノードが高メモリアラームを発し、キューの同期を一時停止します。メモリ使用量は、CloudWatch で RabbitMemUsed および RabbitMqMemLimit のブローカーノードメトリクスを比較することで確認できます。同期は、メッセージが消費もしくは削除される、またはバッチ内のメッセージの数が減るまで完了できません。

> **Note**
>
> キューの同期のバッチサイズを小さくすると、レプリケーショントランザクション数の増加につながる可能性があります。

一時停止されたキューの同期を解決するには、ha-sync-batch-size ポリシーの適用とキューの同期の再開について説明する、このチュートリアルのステップに従ってください。

トピック

- 前提条件
- ステップ 1: ha-sync-batch-size ポリシーを適用する
- ステップ 2: キューの同期を再開する
- 次の手順
- 関連リソース

前提条件

このチュートリアルには、管理者権限を持つ Amazon MQ for RabbitMQ ブローカーユーザーが必要です。ブローカーを初めて作成したときに作成された管理者ユーザー、またはその後で作成した別のユーザーを使用できます。以下の表は、正規表現 (regexp) パターンとしての必要な管理者ユーザータグと許可です。

タグ	読み込み regexp	設定 regexp	書き込み regexp
`administrator`	`.*`	`.*`	`.*`

RabbitMQ ユーザーの作成、およびユーザータグと許可の管理の詳細については、「Amazon MQ for RabbitMQ ブローカーのユーザー」を参照してください。

ステップ 1: **ha-sync-batch-size** ポリシーを適用する

以下の手順では、ブローカーで作成されたすべてのキューに適用されるポリシーの追加について説明します。RabbitMQ ウェブコンソールまたは RabbitMQ Management API を使用できます。詳細については、RabbitMQ ウェブサイトの「Management Plugin」を参照してください。

RabbitMQ ウェブコンソールを使用して **ha-sync-batch-size** ポリシーを適用する

1. Amazon MQ コンソールにサインインします。
2. 左側のナビゲーションペインで [Brokers] (ブローカー) をクリックします。
3. ブローカーのリストから、新しいポリシーを適用するブローカーの名前を選択します。
4. ブローカーのページの [Connections] (接続) セクションで、RabbitMQ ウェブコンソール URL をメモします。新しいブラウザタブまたはウィンドウに RabbitMQ ウェブコンソールが開きます。
5. ブローカー管理者のサインイン認証情報を使用して RabbitMQ ウェブコンソールにログインします。
6. RabbitMQ ウェブコンソールのページ上部で、[Admin] (管理) をクリックします。
7. [Admin] (管理) ページの右側にあるナビゲーションペインで [Policies] (ポリシー) をクリックします。
8. [Policies] (ポリシー) ページに、ブローカーの現在の [User policies] (ユーザーポリシー) が表示されます。[User policies] (ユーザーポリシー) の下で、[Add / update a policy] (ポリシーの追加/更新) を展開します。

> **Note**
>
> デフォルトで、Amazon MQ for RabbitMQ クラスターは、`ha-all-AWS-OWNED-DO-NOT-DELETE` という名前の初期ブローカーポリシーを使用して作成されます。Amazon MQ はこのポリシーを管理して、ブローカー上のすべてのキューが 3 つのノードすべてにレプリケートされ、キューが自動的に同期化されることを確実にします。

9. 新しいブローカーポリシーを作成するには、[Add / update a policy] (ポリシーの追加/更新) で以下を実行します。

   a. [Name] (名前) には、ポリシーの名前 (**`batch-size-policy`** など) を入力します。

   b. [Pattern] (パターン) には regexp パターン **`.*`** を入力して、ポリシーがブローカー上のすべてのキューと一致するようにします。

   c. [Apply to] (適用先) には、ドロップダウンリストから [Exchanges and queues] (エクスチェンジとキュー) を選択します。

   d. [Priority] (優先順位) には、vhost に適用されたその他すべてのポリシーよりも大きい整数を入力します。RabbitMQ のキューとエクスチェンジに適用できるのは、常に 1 つのポリシー定義セットのみです。RabbitMQ は、一致するポリシーで、最高の優先順位値を持つものを選択します。ポリシーの優先順位とポリシーの結合方法の詳細については、RabbitMQ サーバードキュメントの「Policies」を参照してください。

   e. [Definition] (定義) には、以下のキーバリューペアを追加します。

      - **`ha-sync-batch-size`**=*`100`*。ドロップダウンリストから [Number] (数値) を選択します。

        > **Note**
        >
        > `ha-sync-batch-size` の値は、キュー内の同期されていないメッセージの数とサイズに基づいて調整と較正を行う必要がある場合があります。

      - **`ha-mode=all`**。ドロップダウンリストから [String] (文字列) を選択します。

        > **Important**
        >
        > `ha-mode` 定義は、すべての HA 関連ポリシーに必須です。省略すると、検証が失敗します。

- **`ha-sync-mode=automatic`**。ドロップダウンリストから [String] (文字列) を選択します。

> **Note**
>
> `ha-sync-mode` 定義は、すべてのカスタムポリシーに必須です。省略すると、Amazon MQ が定義を自動的に付加します。

f. [Add / update policy] (ポリシーを追加/更新) をクリックします。

10. [User policies] (ユーザーポリシー) リストに新しいポリシーが表示されることを確認します。

RabbitMQ Management API を使用して **`ha-sync-batch-size`** ポリシーを適用する

1. Amazon MQ コンソールにサインインします。
2. 左側のナビゲーションペインで [Brokers] (ブローカー) をクリックします。
3. ブローカーのリストから、新しいポリシーを適用するブローカーの名前を選択します。
4. ブローカーのページの [Connections] (接続) セクションで、RabbitMQ ウェブコンソール URL をメモします。これは、HTTP リクエストで使用するブローカーエンドポイントです。
5. 任意の新しいターミナルまたはコマンドラインウィンドウを開きます。
6. 新しいブローカーポリシーを作成するには、以下の `curl` コマンドを入力します。このコマンドでは、`%2F` としてエンコードされているデフォルト / vhost 上のキューを前提としています。

> **Note**
>
> *`#####`*と*`#####`*を、ブローカー管理者のサインイン認証情報に置き換えます。`ha-sync-batch-size` の値 (*`100`*) は、キュー内の同期されていないメッセージの数とサイズに基づいて調整と較正を行う必要がある場合があります。ブローカーエンドポイントを先ほどメモした URL に置き換えます。

```
curl -i -u username:password -H "content-type:application/json" -XPUT \
-d '{"pattern":".*", "priority":1, "definition":{"ha-sync-batch-size":100, "ha-
mode":"all", "ha-sync-mode":"automatic"}}' \
https://b-589c045f-f8ln-4ab0-a89c-co62e1c32ef8.mq.us-west-2.amazonaws.com/api/
policies/%2F/batch-size-policy
```

7. 新しいポリシーがブローカーのユーザーポリシーに追加されていることを確認するには、以下の curl コマンドを入力して、すべてのブローカーポリシーをリストします。

```
curl -i -u username:password https://b-589c045f-f8ln-4ab0-a89c-co62e1c32ef8.mq.us-west-2.amazonaws.com/api/policies
```

ステップ 2: キューの同期を再開する

ブローカーに新しい ha-sync-batch-size ポリシーを適用したら、キューの同期を再開します。

RabbitMQ ウェブコンソールを使用してキューの同期を再開する

> **Note**
>
> RabbitMQ ウェブコンソールを開くには、このチュートリアルのステップ 1 にある前述の手順を参照してください。

1. RabbitMQ ウェブコンソールのページ上部で、[Queues] (キュー) をクリックします。
2. [Queues] (キュー) ページの [All queues] (すべてのキュー) で、一時停止されたキューを見つけます。[ポリシー] 行に、キューが作成された新しいポリシーの名前をリストします (batch-size-policy など)。
3. 縮小されたバッチサイズで同期プロセスを再開するには、まずキュー同期をキャンセルします。次に、キュー同期を再起動します。

> **Note**
>
> 同期が一時停止して正常に終了しない場合は、ha-sync-batch-size の値を低くして、もう一度キューの同期を再開してみてください。

次の手順

- キューが正常に同期化されたら、Amazon CloudWatch メトリクス RabbitMQMemUsed を表示することで、RabbitMQ ノードが使用するメモリの量をモニタリングできます。RabbitMQMemLimit メトリクスを表示して、ノードのメモリ制限をモニタリングすることもできます。詳細については、「Amazon MQ 向けの CloudWatch メトリクスへのアクセス」および

「Amazon MQ for RabbitMQ ブローカーで利用可能な CloudWatch メトリクス」を参照してください。

- キューの同期が一時停止しないようにするため、キューを短くしておき、メッセージを処理することをお勧めします。メッセージサイズが大きいワークロードの場合は、より多くのメモリを備えたより大きなインスタンスサイズにブローカーインスタンスタイプをアップグレードすることもお勧めします。ブローカーインスタンスタイプとブローカー設定の編集に関する詳細については、「ブローカー設定の編集」を参照してください。
- 新しい Amazon MQ for RabbitMQ ブローカーを作成するときは、ブローカーのパフォーマンスを最適化するために、Amazon MQ が一連のデフォルトブローカーポリシーと仮想ホスト制限を適用します。お使いのブローカーに推奨されるデフォルトのポリシーと制限がない場合は、独自のポリシーと制限を作成することをお勧めします。デフォルトのポリシーと vhost 制限の作成に関する詳細については、「https://docs.aws.amazon.com//amazon-mq/latest/developer-guide/rabbitmq-defaults.html」を参照してください。

関連リソース

- UpdateBrokerInput – Amazon MQ API を使用してブローカーインスタンスタイプを更新するには、このブローカープロパティを使用します。
- Parameters and Policies (RabbitMQ サーバードキュメント) – RabbitMQ のウェブサイトで、RabbitMQ のパラメータとポリシーの詳細について学びます。
- RabbitMQ Management HTTP API – RabbitMQ Management API の詳細について学びます。

接続およびチャネルの数の削減

Amazon MQ 上の RabbitMQ への接続は、クライアントアプリケーションで終了できます。また、RabbitMQ ウェブコンソールを使用して手動で終了することもできます。RabbitMQ ウェブコンソールを使用して接続を終了するには、次の手順を実行します。

1. にサインイン AWS マネジメントコンソール し、ブローカーの RabbitMQ ウェブコンソールを開きます。
2. RabbitMQ コンソールで、[Connections] (接続) タブを選択します。
3. [Connections] (接続) ページの [All connections] (すべての接続) から、終了する接続の名前をリストから選択します。
4. 接続の詳細ページで、[Close this connection] (この接続を終了する) を選択してセクションを展開し、[Force Close] (強制終了) を選択します。オプションで、理由のデフォルトのテキス

トをお客様自身の説明に置き換えることもできます。接続を終了すると、Amazon MQ 上の RabbitMQ により、指定した理由がクライアントに返されます。

5. ダイアログボックスで [OK] を選択し、確認して接続を終了します。

接続を終了すると、終了した接続に関連付けられているすべてのチャネルも終了します。

> **Note**
>
> クライアントアプリケーションは、終了後にブローカーが自動的に接続を再確立するように設定されている場合があります。この場合、接続またはチャネルの数を減らすには、ブローカーのウェブコンソールからの接続を終了するだけでは不十分です。

パブリックアクセスがないブローカーの場合、適切なメッセージプロトコルのポート (例えば AMQP 接続の場合、ポート 5671) でインバウンドトラフィックを拒否することで、一時的に接続をブロックできます。ブローカーの作成時に Amazon MQ に指定したセキュリティグループのポートをブロックできます。セキュリティグループの変更方法の詳細については、Amazon VPC ユーザーガイドの「セキュリティグループへのルールの追加」を参照してください。

ステップ 2: ブローカーに JVM ベースのアプリケーションを接続する

RabbitMQ ブローカーを作成したら、ブローカーにアプリケーションを接続できます。以下の例では、RabbitMQ Java クライアントライブラリを使用してブローカーへの接続を作成し、キューを作成して、メッセージを送信する方法を説明します。RabbitMQ ブローカーには、サポートされているさまざまな言語の RabbitMQ クライアントライブラリを使用して接続することができます。サポートされている RabbitMQ クライアントライブラリの詳細については、「RabbitMQ client libraries and developer tools」を参照してください。

前提条件

> **Note**
>
> 以下の前提条件ステップは、パブリックアクセシビリティなしで作成された RabbitMQ ブローカーのみに適用されます。パブリックアクセシビリティがあるブローカーを作成している場合は、スキップすることができます。

VPC 属性 を有効にする

VPC 内でブローカーにアクセスできることを確実にするには、`enableDnsHostnames` および `enableDnsSupport` VPC 属性を有効にする必要があります。詳細については、Amazon VPC ユーザーガイドの「VPC の DNS サポート」を参照してください。

インバウンド接続を有効にする

1. Amazon MQ コンソールにサインインします。
2. ブローカーのリストからブローカーの名前 (MyBroker など) を選択します。
3. [***MyBroker***] ページの [Connections] (接続) セクションで、ブローカーのウェブコンソール URL とワイヤレベルプロトコルのアドレスとポートをメモします。
4. [Details] (詳細) セクションの [Security and network] (セキュリティとネットワーク) で、セキュリティグループの名前または
   
   をクリックします。

   EC2 ダッシュボードの [セキュリティグループ] ページが表示されます。
5. セキュリティグループのリストから、セキュリティグループを選択します。
6. ページ下部で、[インバウンド] を選択し、次に [編集] を選択します。
7. [Edit inbound rules] (インバウンドルールの編集) ダイアログボックスで、パブリックアクセスを許可する URL またはエンドポイントごとにルールを追加します (以下の例は、これをブローカーのウェブコンソールに対して行う方法を説明しています)。

   a. [ルールの追加] を選択します。

   b. [タイプ] で、[カスタム TCP] を選択します。

   c. [Source] (ソース) では、[Custom] (カスタム) が選択された状態のままにしておき、ウェブコンソールにアクセスできるようにするシステムの IP アドレスを入力します (`192.0.2.1` など)。

   d. [Save] (保存) をクリックします。

      これで、ブローカーはインバウンド接続を受け入れることができます。

Java の依存関係を追加する

ビルドの自動化のために Apache Maven を使用している場合は、以下の依存関係を pom.xml ファイルに追加します。Apache Maven のプロジェクトオブジェクトモデルファイルの詳細については、「Introduction to the POM」を参照してください。

```
<dependency>
    <groupId>com.rabbitmq</groupId>
    <artifactId>amqp-client</artifactId>
    <version>5.9.0</version>
</dependency>
```

ビルドの自動化のために Gradle を使用している場合は、以下の依存関係を宣言します。

```
dependencies {
    compile 'com.rabbitmq:amqp-client:5.9.0'
}
```

Connection と **Channel** クラスをインポートする

RabbitMQ Java クライアントは、そのトップレベルパッケージとして com.rabbitmq.client を使用し、それぞれが AMQP 0-9-1 接続とチャネルを表す Connection および Channel API クラスがあります。以下の例にあるように、使用する前に Connection と Channel クラスをインポートします。

```
import com.rabbitmq.client.Connection;
import com.rabbitmq.client.Channel;
```

ConnectionFactory を作成してブローカーに接続する

以下の例を使用して、所定のパラメータで ConnectionFactory クラスのインスタンスを作成します。setHost メソッドを使用して、先ほどメモしておいたブローカーエンドポイントを設定します。AMQPS のワイヤレベル接続には、ポート 5671 を使用します。

```
ConnectionFactory factory = new ConnectionFactory();

factory.setUsername(username);
factory.setPassword(password);

//Replace the URL with your information
```

```
factory.setHost("b-c8352341-ec91-4a78-ad9c-a43f23d325bb.mq.us-west-2.amazonaws.com");
factory.setPort(5671);

// Allows client to establish a connection over TLS
factory.useSslProtocol();

// Create a connection
Connection conn = factory.newConnection();

// Create a channel
Channel channel = conn.createChannel();
```

エクスチェンジにメッセージを発行する

エクスチェンジにメッセージを発行するには、Channel.basicPublish を使用できます。以下の例では、AMQP Builder クラスを使用して、content-type が plain/text のメッセージプロパティオブジェクトを構築します。

```
byte[] messageBodyBytes = "Hello, world!".getBytes();
channel.basicPublish(exchangeName, routingKey,
             new AMQP.BasicProperties.Builder()
               .contentType("text/plain")
               .userId("userId")
               .build(),
               messageBodyBytes);
```

> **Note**
>
> BasicProperties は自動生成されたホルダークラス AMQP の内部クラスであることに注意してください。

キューにサブスクライブしてメッセージを受信する

メッセージは、Consumer インターフェイスを使用してキューにサブスクライブすることによって受信できます。サブスクライブすると、メッセージが到着すると同時に自動配信されます。

Consumer を実装する最も簡単な方法は、サブクラス DefaultConsumer の使用です。以下の例にあるように、DefaultConsumer オブジェクトは、サブスクリプションをセットアップするための basicConsume コールの一部として渡すことがきます。

```
boolean autoAck = false;
channel.basicConsume(queueName, autoAck, "myConsumerTag",
     new DefaultConsumer(channel) {
         @Override
         public void handleDelivery(String consumerTag,
                                    Envelope envelope,
                                    AMQP.BasicProperties properties,
                                    byte[] body)
             throws IOException
         {
             String routingKey = envelope.getRoutingKey();
             String contentType = properties.getContentType();
             long deliveryTag = envelope.getDeliveryTag();
             // (process the message components here ...)
             channel.basicAck(deliveryTag, false);
         }
     });
```

> **Note**
>
> autoAck = false を指定したので、Consumer に配信されたメッセージを承認する必要があります。これは、上記の例にあるように、handleDelivery で実行することが最も便利です。

接続を閉じてブローカーへの接続を切断する

RabbitMQ ブローカーへの接続を切断するには、以下に示すように、チャネルと接続の両方を閉じます。

```
channel.close();
conn.close();
```

> **Note**
>
> RabbitMQ Java クライアントライブラリの使用に関する詳細については、RabbitMQ Java Client API Guide を参照してください。

ステップ 3: (オプション) AWS Lambda 関数に接続する

AWS Lambda は Amazon MQ ブローカーに接続して、Amazon MQ ブローカーからのメッセージを消費できます。ブローカーを Lambda に接続するときは、キューからメッセージを読み取り、関数 synchronously を呼び出すイベントソースマッピングを作成します。作成するイベントソースマッピングは、ブローカーからメッセージをバッチで読み取り、それらを JSON オブジェクト形式の Lambda ペイロードに変換します。

ブローカーを Lambda 関数に接続する

1. Lambda 関数 execution role に以下の IAM ロール許可を追加します。

   - mq:DescribeBroker
   - ec2:CreateNetworkInterface
   - ec2:DeleteNetworkInterface
   - ec2:DescribeNetworkInterfaces
   - ec2:DescribeSecurityGroups
   - ec2:DescribeSubnets
   - ec2:DescribeVpcs
   - logs:CreateLogGroup
   - logs:CreateLogStream
   - logs:PutLogEvents
   - secretsmanager:GetSecretValue

   > **Note**
   >
   > 必要な IAM 許可がない場合、関数は Amazon MQ リソースからレコードを正常に読み取ることができません。

2. (オプション) パブリックアクセシビリティがないブローカーを作成した場合は、次のいずれかを実行して、Lambda のブローカーへの接続を許可する必要があります。

   - パブリックサブネットごとに 1 つの NAT ゲートウェイを設定します。詳細については、AWS Lambda デベロッパーガイドの「VPC に接続した関数のインターネットアクセスとサービスアクセス」を参照してください。

- VPC エンドポイントを使用して、Amazon Virtual Private Cloud (Amazon VPC) と Lambda 間の接続を作成します。Amazon VPC は、 AWS Security Token Service (AWS STS) および Secrets Manager エンドポイントにも接続する必要があります。詳細については、AWS Lambda デベロッパーガイドの「Lambda のインターフェイス VPC エンドポイントの設定」を参照してください。

3. AWS マネジメントコンソールを使用して、Lambda 関数のイベントソースとしてブローカーを設定します。`create-event-source-mapping` AWS Command Line Interface コマンドを使用することもできます。

4. ブローカーから取り込まれたメッセージを処理するための Lambda 関数のコードをいくつか記述します。イベントソースマッピングによって取得される Lambda ペイロードは、ブローカーのエンジンタイプに依存します。以下は、Amazon MQ for RabbitMQ キューの Lambda ペイロードの例です。

> **Note**
>
> この例では、`test` がキューの名前で、`/` がデフォルト仮想ホストの名前です。メッセージを受信すると、イベントソースは `test::/` の下にメッセージを一覧表示します。

```
{
  "eventSource": "aws:rmq",
  "eventSourceArn": "arn:aws:mq:us-
west-2:112556298976:broker:test:b-9bcfa592-423a-4942-879d-eb284b418fc8",
  "rmqMessagesByQueue": {
    "test::/": [
      {
        "basicProperties": {
          "contentType": "text/plain",
          "contentEncoding": null,
          "headers": {
            "header1": {
              "bytes": [
                118,
                97,
                108,
                117,
                101,
                49
```

```
            ]
          },
          "header2": {
            "bytes": [
              118,
              97,
              108,
              117,
              101,
              50
            ]
          },
          "numberInHeader": 10
        }
        "deliveryMode": 1,
        "priority": 34,
        "correlationId": null,
        "replyTo": null,
        "expiration": "60000",
        "messageId": null,
        "timestamp": "Jan 1, 1970, 12:33:41 AM",
        "type": null,
        "userId": "AIDACKCEVSQ6C2EXAMPLE",
        "appId": null,
        "clusterId": null,
        "bodySize": 80
      },
      "redelivered": false,
      "data": "eyJ0aW1lb3V0IjowLCJkYXRhIjoiQ1pybWYwR3c4T3Y0YnFMUXhENEUifQ=="
    }
  ]
 }
}
```

Amazon MQ の Lambda への接続、Amazon MQ イベントソースに対して Lambda がサポートするオプション、およびイベントソースマッピングエラーに関する詳細については、「AWS Lambda デベロッパーガイド」の「Amazon MQ で Lambda を使用する」を参照してください。

Amazon MQ for RabbitMQ での OAuth 2.0 認証と認可の使用

このチュートリアルでは、Amazon Cognito を OAuth 2.0 プロバイダーとして使用して、Amazon MQ for RabbitMQ ブローカーの OAuth 2.0 認証を設定する方法について説明します。

> **Note**
>
> Amazon Cognito は、中国 (北京) および中国 (寧夏) では使用できません。

> **Important**
>
> このチュートリアルは Amazon Cognito に固有のものですが、他の ID プロバイダー (IdP) を使用できます。詳細については、「OAuth 2.0 認証の例」を参照してください。

このページの内容

OAuth 2.0 認証を設定するための前提条件

このチュートリアルで必要な Amazon Cognito リソースを設定するには、AWS CDK スタックである RabbitMQ OAuth 2 プラグイン用 Amazon Cognito スタックをデプロイします。Amazon Cognito を手動で設定する場合は、Amazon MQ for RabbitMQ ブローカーで OAuth 2.0 を設定する前に、次の前提条件を満たしていることを確認してください。

Amazon Cognito をセットアップするための前提条件

- ユーザープールを作成して Amazon Cognito エンドポイントをセットアップします。これを行うには、「How to use OAuth 2.0 in Amazon Cognito: Learn about the different OAuth 2.0 grants」というタイトルのブログを参照してください。
- ユーザープールに `rabbitmq` という名前のリソースサーバーを作成し、`read:all`、`write:all`、`configure:all`、`tag:administrator` のスコープを定義します。これらのスコープは RabbitMQ アクセス許可に関連付けられます。

  リソースサーバーの作成の詳細については、「Amazon Cognito デベロッパーガイド」の「ユーザープール (AWS マネジメントコンソール) のリソースサーバーの定義」を参照してください。
- 以下のアプリケーションクライアントを作成します。
  - タイプ `Machine-to-Machine application` のユーザープールのアプリケーションクライアント。これは、RabbitMQ AMQP クライアントに使用されるクライアントシークレットを持つ機

密クライアントです。アプリケーションクライアントとその作成の詳細については、「アプリクライアントの種類」および「アプリクライアントの作成」を参照してください。

- タイプ `Single-page application` のユーザープールのアプリケーションクライアント。これは、RabbitMQ マネジメントコンソールへのユーザーのログインに使用されるパブリッククライアントです。このアプリケーションクライアントを更新して、次の手順で作成する Amazon MQ for RabbitMQ ブローカーのエンドポイントを許可されたコールバック URL として含める必要があります。詳細については、「Amazon Cognito コンソールでのマネージドログインのセットアップ」を参照してください。

Amazon MQ をセットアップするための前提条件

- OAuth 2.0 のセットアップが成功したかどうかを確認する bash スクリプトを実行するための作業中の Docker インストール。
- ブローカーの作成中にユーザー名とパスワードの追加をオプションにする AWS CLI version >= `2.28.23`。

AWS CLI を使用した Amazon Cognito による OAuth 2.0 認証の設定

次の手順は、Amazon Cognito を IdP として使用して Amazon MQ for RabbitMQ ブローカーの OAuth 2.0 認証を設定する方法を示しています。この手順では、AWS CLI を使用して必要なリソースを作成および設定します。

次の手順では、configurationID や Revision、*<c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca>*、*<2>* などのプレースホルダー値を実際の値に置き換えてください。

1. 次の例に示すように、create-configuration AWS CLI コマンドを使用して新しい設定を作成します。

```
aws mq create-configuration \
  --name "rabbitmq-oauth2-config" \
  --engine-type "RABBITMQ" \
  --engine-version "3.13"
```

このコマンドでは、次の例のようなレスポンスが返されます。

```
{
    "Arn": "arn:aws:mq:us-west-2:123456789012:configuration:c-fa3390a5-7e01-4559-
ae0c-eb15b38b22ca",
```

```
    "AuthenticationStrategy": "simple",
    "Created": "2025-07-17T16:03:01.759943+00:00",
    "Id": "c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca",
    "LatestRevision": {
    "Created": "2025-07-17T16:03:01.759000+00:00",
    "Description": "Auto-generated default for rabbitmq-oauth2-config on RabbitMQ
 3.13",
    "Revision": 1
    },
    "Name": "rabbitmq-oauth2-config"
}
```

2. 次の例に示すように、OAuth 2.0 を認証および認可方法として使用する **rabbitmq.conf** と呼ばれる設定ファイルを作成します。

```
auth_backends.1 = oauth2

# FIXME: Update this value with the token signing key URL of your Amazon Cognito
 user pool.
# If you used the AWS CDK stack to deploy Amazon Cognito, this is one of the stack
 outputs.
auth_oauth2.jwks_url = ${RabbitMqOAuth2TestStack.JwksUri}
auth_oauth2.resource_server_id = rabbitmq
# Amazon Cognito does not include an audience field in access tokens
auth_oauth2.verify_aud = false

# Amazon Cognito does not allow * in its custom scopes. Use aliases to translate
 between Amazon Cognito and RabbitMQ.
auth_oauth2.scope_prefix = rabbitmq/
auth_oauth2.scope_aliases.1.alias = rabbitmq/read:all
auth_oauth2.scope_aliases.1.scope = rabbitmq/read:*/*
auth_oauth2.scope_aliases.2.alias = rabbitmq/write:all
auth_oauth2.scope_aliases.2.scope = rabbitmq/write:*/*
auth_oauth2.scope_aliases.3.alias = rabbitmq/configure:all
auth_oauth2.scope_aliases.3.scope = rabbitmq/configure:*/*

# Allow OAuth 2.0 login for RabbitMQ management console
management.oauth_enabled = true
# FIXME: Update this value with the client ID of your public application client
management.oauth_client_id
 = ${RabbitMqOAuth2TestStack.ManagementConsoleAppClientId}
# FIXME: Update this value with the base JWKS URI (without /.well-known/jwks.json)
auth_oauth2.issuer = ${RabbitMqOAuth2TestStack.Issuer}
```

```
management.oauth_scopes = rabbitmq/tag:administrator
```

この設定では、スコープエイリアスを使用して、Amazon Cognito で定義されたスコープを RabbitMQ 互換スコープにマッピングします。

3. 次の例に示すように、update-configuration AWS CLI コマンドを使用して設定を更新します。このコマンドでは、この手順のステップ 1 のレスポンスで受け取った設定 ID を追加します。例えば、**c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca**。

```
aws mq update-configuration \
  --configuration-id "<c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca>" \
  --data "$(cat rabbitmq.conf | base64 --wrap=0)"
```

このコマンドでは、次の例のようなレスポンスが返されます。

```
{
    "Arn": "arn:aws:mq:us-west-2:123456789012:configuration:c-b600ac8e-8183-4f74-
a713-983e59f30e3d",
    "Created": "2025-07-17T16:57:04.520931+00:00",
    "Id": "c-b600ac8e-8183-4f74-a713-983e59f30e3d",
    "LatestRevision": {
        "Created": "2025-07-17T16:57:39.172000+00:00",
        "Revision": 2
    },
    "Name": "rabbitmq-oauth2-config",
    "Warnings": []
}
```

4. この手順のステップ 2 で作成した OAuth 2.0 設定でブローカーを作成します。そのためには、以下の例に示すように create-broker AWS CLI コマンドを使用します。このコマンドでは、ステップ 1 と 2 のレスポンスで取得した設定 ID とリビジョン番号をそれぞれ指定します。例えば、**c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca** と **2** です。

```
aws mq create-broker \
 --broker-name "rabbitmq-oauth2-broker" \
 --engine-type "RABBITMQ" \
 --engine-version "3.13" \
 --host-instance-type "mq.m7g.large" \
 --deployment-mode "CLUSTER_MULTI_AZ" \
 --logs '{"General": true}' \
 --publicly-accessible \
```

```
 --configuration '{"Id": "<c-fa3390a5-7e01-4559-ae0c-
eb15b38b22ca>","Revision": <2>}' \
```

このコマンドでは、次の例のようなレスポンスが返されます。

```
{
    "BrokerArn": "arn:aws:mq:us-west-2:123456789012:broker:rabbitmq-oauth2-
broker:b-2a1b5133-a10c-49d2-879b-8c176c34cf73",
    "BrokerId": "b-2a1b5133-a10c-49d2-879b-8c176c34cf73"
}
```

5. 次の例に示すように、describe-broker AWS CLI コマンドを使用して、ブローカーのステータスが CREATION_IN_PROGRESS から RUNNING に移行していることを確認します。このコマンドでは、前のステップの結果で取得したブローカー ID を指定します。例: **b-2a1b5133-a10c-49d2-879b-8c176c34cf73**。

```
aws mq describe-broker \
 --broker-id "<b-2a1b5133-a10c-49d2-879b-8c176c34cf73>"
```

このコマンドでは、次の例のようなレスポンスが返されます。次のレスポンスは、describe-broker コマンドが返す完全な出力の省略バージョンです。このレスポンスは、ブローカーのステータスと、ブローカーの保護に使用される認証戦略を示します。この場合、config_managed 認証戦略はブローカーが OAuth 2 認証方法を使用していることを示します。

```
{
    "AuthenticationStrategy": "config_managed",
    ...,
    "BrokerState": "RUNNING",
    ...
}
```

OAuth2 を使用して RabbitMQ マネジメントコンソールにログインするには、ブローカーエンドポイントを、対応する Amazon Cognito アプリクライアントの有効なコールバック URL として追加する必要があります。詳細については、サンプル Amazon Cognito CDK スタックのセットアップのステップ 5 を参照してください。

6. 次の perf-test.sh スクリプトを使用して、OAuth 2.0 の認証と認可を検証します。

この bash スクリプトを使用して、Amazon MQ for RabbitMQ ブローカーへの接続をテストします。このスクリプトは Amazon Cognito からトークンを取得し、接続が正しく設定されているかどうかを確認します。正常に設定されると、ブローカーがメッセージを発行して消費します。

ACCESS_REFUSED エラーが発生した場合は、ブローカーの CloudWatch ログを使用して設定をトラブルシューティングできます。ブローカーの CloudWatch ロググループのリンクは、Amazon MQ コンソールにあります。

このスクリプトでは、次の値を指定する必要があります。

- CLIENT_ID および CLIENT_SECRET: これらの値は、Amazon Cognito コンソールの [アプリクライアント] ページで確認できます。
- Cognito ドメイン: これは Amazon Cognito コンソールで確認できます。[ブランディング] で、[ドメイン] を選択します。[ドメイン] ページで、この値は [リソースサーバー] セクションにあります。
- Amazon MQ ブローカーエンドポイント: この値は、Amazon MQ コンソールのブローカーの詳細ページの [接続] にあります。

```
#! /bin/bash
set -e

# Client information
## FIXME: Update this value with the client ID and secret of your confidential
 application client
CLIENT_ID=${RabbitMqOAuth2TestStack.AmqpAppClientId}
CLIENT_SECRET=${RabbitMqOAuth2TestStack.AmqpAppClientSecret}

# FIXME: Update this value with the domain of your Amazon Cognito user pool
RESPONSE=$(curl -X POST ${RabbitMqOAuth2TestStack.TokenEndpoint} \
                -H "Content-Type: application/x-www-form-urlencoded" \
                -d
 "grant_type=client_credentials&client_id=${CLIENT_ID}&client_secret=${CLIENT_SECRET}&scope
configure:all rabbitmq/read:all rabbitmq/tag:administrator rabbitmq/write:all")


# Extract the access_token from the response.
# This token will be passed in the password field when connecting to the broker.
# Note that the username is left blank, the field is ignored by the plugin.
BROKER_PASSWORD=$(echo ${RESPONSE} | jq -r '.access_token')
```

```
# FIXME: Update this value with the endpoint of your broker. For
 example, b-89424106-7e0e-4abe-8e98-8de0dada7630.mq.us-east-1.on.aws.
BROKER_DNS=<broker_dns>
CONNECTION_STRING=amqps://:${BROKER_PASSWORD}@${BROKER_DNS}:5671

# Produce/consume messages using the above connection string
QUEUES_COUNT=1
PRODUCERS_COUNT=1
CONSUMERS_COUNT=1
PRODUCER_RATE=1

docker run -it --rm --ulimit nofile=40960:40960 pivotalrabbitmq/perf-test:latest \
    --queue-pattern 'test-queue-%d' --queue-pattern-from 1 --queue-pattern-to
 $QUEUES_COUNT \
    --producers $PRODUCERS_COUNT --consumers $CONSUMERS_COUNT \
    --id "test${QUEUES_COUNT}q${PRODUCERS_COUNT}p${CONSUMERS_COUNT}c
${PRODUCER_RATE}r" \
    --uri ${CONNECTION_STRING} \
    --flag persistent --rate $PRODUCER_RATE
```

Amazon Cognito による OAuth 2.0 とシンプルな認証の設定

OAuth 2.0 認証を使用してブローカーを作成する場合、次のいずれかの認証方法を指定できます。

- OAuth 2.0 のみ: この方法を使用するには、ブローカーの作成時にユーザー名とパスワードを指定しないでください。前の手順は、OAuth 2.0 認証方法のみを使用する方法を示しています。
- OAuth 2.0 とシンプルな認証の両方: この方法を使用するには、ブローカーの作成時にユーザー名とパスワードを指定します。また、次の手順に示すように、`auth_backends.2 = internal` をブローカー設定に追加します。

次の手順では、*<ConfigurationId>* や *<Revision>* などのプレースホルダー値を実際の値に置き換えてください。

1. 両方の認証方法を使用するには、次の例に示すように、ブローカー設定を作成します。

```
auth_backends.1 = oauth2
auth_backends.2 = internal
```

```
# FIXME: Update this value with the token signing key URL of your Amazon Cognito
 user pool
auth_oauth2.jwks_url = ${RabbitMqOAuth2TestStack.JwksUri}
auth_oauth2.resource_server_id = rabbitmq
auth_oauth2.verify_aud = false

auth_oauth2.scope_prefix = rabbitmq/
auth_oauth2.scope_aliases.1.alias = rabbitmq/read:all
auth_oauth2.scope_aliases.1.scope = rabbitmq/read:*/*
auth_oauth2.scope_aliases.2.alias = rabbitmq/write:all
auth_oauth2.scope_aliases.2.scope = rabbitmq/write:*/*
auth_oauth2.scope_aliases.3.alias = rabbitmq/configure:all
auth_oauth2.scope_aliases.3.scope = rabbitmq/configure:*/*
```

この設定では、スコープエイリアスを使用して、Amazon Cognito で定義されたスコープを RabbitMQ 互換スコープにマッピングします。

2. 次の例に示すように、両方の認証方法を使用するブローカーを作成します。

```
aws mq create-broker \
 --broker-name "rabbitmq-oauth2-broker-with-internal-user" \
 --engine-type "RABBITMQ" \
 --engine-version "3.13" \
 --host-instance-type "mq.m7g.large" \
 --deployment-mode "CLUSTER_MULTI_AZ" \
 --logs '{"General": true}' \
 --publicly-accessible \
 --configuration '{"Id": "<ConfigurationId>","Revision": <Revision>}' \
 --users '[{"Username":"<myUser>","Password":"<myPassword11>"}]'
```

3. Amazon Cognito による OAuth 2.0 認証の設定 手順のステップ 5 と 6 で説明されているように、ブローカーのステータスと認証方法の設定が成功したことを確認します。

Amazon MQ for RabbitMQ での IAM 認証と認可の使用

次の手順は、Amazon MQ for RabbitMQ AWS ブローカーの IAM 認証と認可を有効にする方法を示しています。IAM を有効にすると、ユーザーは IAM AWS 認証情報を使用して RabbitMQ Management API にアクセスし、AMQP 経由で接続するための認証を行うことができます。Amazon MQ for RabbitMQ での IAM 認証の仕組みの詳細については、「」を参照してくださいthe section called “IAM 認証と認可”。

前提条件

- AWS Amazon MQ for RabbitMQ ブローカーを所有する AWS アカウントの管理者認証情報
- これらの管理者認証情報で設定されたシェル環境 (CLI AWS プロファイルまたは環境変数を使用)
- AWS CLI のインストールと設定
- `jq` コマンドライン JSON プロセッサがインストールされている
- `curl` コマンドラインツールのインストール

を使用した IAM 認証と認可の設定 AWS CLI

1. 環境変数を設定する

   ブローカーに必要な環境変数を設定します。

   ```
   export AWS_DEFAULT_REGION=<region>
   export BROKER_ID=<broker-id>
   ```

2. アウトバウンド JWT トークンを有効にする

   AWS アカウントのアウトバウンドウェブ ID フェデレーションを有効にします。

   ```
   ISSUER_IDENTIFIER=$(aws iam enable-outbound-web-identity-federation --query
    'IssuerIdentifier' --output text)
   echo $ISSUER_IDENTIFIER
   ```

   出力には、アカウントの一意の発行者識別子 URL が 形式で表示されます`https://<id>.tokens.sts.global.api.aws`。

3. IAM ポリシードキュメントを作成する

   ウェブ ID トークンを取得するアクセス許可を付与するポリシードキュメントを作成します。

   ```
   cat > policy.json << 'EOF'
   {
       "Version": "2012-10-17",
   ```

```
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "sts:GetWebIdentityToken",
                "sts:TagGetWebIdentityToken"
            ],
            "Resource": "*"
        }
    ]
}
EOF
```

4. 信頼ポリシーを作成する

   発信者 ID を取得し、信頼ポリシードキュメントを作成します。

```
CALLER_ARN=$(aws sts get-caller-identity --query Arn --output text)
cat > trust-policy.json << EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "$CALLER_ARN"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF
```

5. IAM ロールを作成する

   IAM ロールを作成し、ポリシーをアタッチします。

```
aws iam create-role --role-name RabbitMqAdminRole --assume-role-policy-document
 file://trust-policy.json
aws iam put-role-policy --role-name RabbitMqAdminRole --policy-name
 RabbitMqAdminRolePolicy --policy-document file://policy.json
```

6. RabbitMQ OAuth2 設定を構成する

   OAuth2 認証および認可設定を使用して RabbitMQ 設定ファイルを作成します。

```
cat > rabbitmq.conf << EOF
auth_backends.1 = oauth2
auth_backends.2 = internal

auth_oauth2.jwks_url = ${ISSUER_IDENTIFIER}/.well-known/jwks.json
auth_oauth2.resource_server_id = rabbitmq
auth_oauth2.scope_prefix = rabbitmq/

auth_oauth2.additional_scopes_key = sub
auth_oauth2.scope_aliases.1.alias = arn:aws:iam::$(aws sts get-caller-identity --
query Account --output text):role/RabbitMqAdminRole
auth_oauth2.scope_aliases.1.scope = rabbitmq/tag:administrator rabbitmq/read:*/*
 rabbitmq/write:*/* rabbitmq/configure:*/*
auth_oauth2.https.hostname_verification = wildcard

management.oauth_enabled = true
EOF
```

7. ブローカー設定を更新する

   新しい設定をブローカーに適用します。

```
# Retrieve the configuration ID
CONFIG_ID=$(aws mq describe-broker --broker-id $BROKER_ID --query
 'Configurations[0].Id' --output text)

# Create a new configuration revision
REVISION=$(aws mq update-configuration --configuration-id $CONFIG_ID --data "$(cat
 rabbitmq.conf | base64 --wrap=0)" --query 'LatestRevision.Revision' --output text)
```

```
# Apply the configuration to the broker
aws mq update-broker --broker-id $BROKER_ID --configuration Id=$CONFIG_ID,Revision=
$REVISION

# Reboot the broker to apply changes
aws mq reboot-broker --broker-id $BROKER_ID
```

ブローカーのステータスが に戻るのを待ってRUNNINGから、次のステップに進みます。

8. JWT トークンを取得する

   IAM ロールを引き受け、ウェブ ID トークンを取得します。

```
# Assume the RabbitMqAdminRole
ROLE_CREDS=$(aws sts assume-role --role-arn arn:aws:iam::$(aws sts get-caller-
identity --query Account --output text):role/RabbitMqAdminRole --role-session-name
 rabbitmq-session)

# Configure the session with temporary credentials
export AWS_ACCESS_KEY_ID=$(echo "$ROLE_CREDS" | jq -r '.Credentials.AccessKeyId')
export AWS_SECRET_ACCESS_KEY=$(echo "$ROLE_CREDS" | jq -r
 '.Credentials.SecretAccessKey')
export AWS_SESSION_TOKEN=$(echo "$ROLE_CREDS" | jq -r '.Credentials.SessionToken')

# Obtain the web identity token
TOKEN_RESPONSE=$(aws sts get-web-identity-token \
    --audience "rabbitmq" \
    --signing-algorithm ES384 \
    --duration-seconds 300 \
    --tags Key=scope,Value="rabbitmq/tag:administrator")

# Extract the token
TOKEN=$(echo "$TOKEN_RESPONSE" | jq -r '.WebIdentityToken')
```

9. RabbitMQ 管理 API にアクセスする

   JWT トークンを使用して RabbitMQ Management API にアクセスします。

```
BROKER_URL=<broker-id>.mq.<region>.on.aws
```

```
curl -u ":$TOKEN" \
    -X GET https://${BROKER_URL}/api/overview \
    -H "Content-Type: application/json"
```

応答が成功すると、IAM 認証が正しく動作していることが確認されます。レスポンスには、JSON 形式のブローカーの概要情報が含まれています。

10. JWT トークンを使用して AMQP 経由で接続する

    パフォーマンステストツールで JWT トークンを使用して AMQP 接続をテストします。

```
BROKER_DNS=<broker-endpoint>
CONNECTION_STRING=amqps://:${TOKEN}@${BROKER_DNS}:5671

docker run -it --rm --ulimit nofile=40960:40960 pivotalrabbitmq/perf-test:latest \
    --queue-pattern 'test-queue-%d' --queue-pattern-from 1 --queue-pattern-to 1 \
    --producers 1 --consumers 1 \
    --uri ${CONNECTION_STRING} \
    --flag persistent --rate 1
```

    ACCESS_REFUSED エラーが発生した場合は、ブローカーの CloudWatch ログを使用して設定をトラブルシューティングできます。ブローカーの CloudWatch Logs ロググループのリンクは、Amazon MQ コンソールにあります。

Amazon MQ for RabbitMQ での LDAP 認証と認可の使用

このチュートリアルでは、 を使用して Amazon MQ for RabbitMQ ブローカーの LDAP 認証と認可を設定する方法について説明します AWS Managed Microsoft AD。

このページの内容

LDAP 認証と認可を設定するための前提条件

このチュートリアルで必要な AWS リソースを設定するには、AWS Amazon MQ for RabbitMQ LDAP 統合用の CDK スタックを と AWS Managed Microsoft ADデプロイします。

この CDK スタックは、LDAP ユーザーとグループ AWS Managed Microsoft AD、Network Load Balancer、証明書、IAM ロールなど、必要なすべての AWS リソースを自動的に作成します。スタックによって作成されたリソースの完全なリストについては、パッケージ README を参照してください。

CDK スタックを使用する代わりにリソースを手動で設定する場合は、Amazon MQ for RabbitMQ ブローカーで LDAP を設定する前に、同等のインフラストラクチャが整っていることを確認してください。

Amazon MQ をセットアップするための前提条件

AWS CLI バージョン >= 2.28.23。ブローカーの作成時にユーザー名とパスワードをオプションで追加します。

CLI を使用した RabbitMQ での LDAP AWS の設定

この手順では、 CLI AWS を使用して必要なリソースを作成および設定します。次の手順では、configurationID や Revision、 <c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca> などのプレースホルダー値を実際の値に置き換え<2>てください。

1. 次の例に示すように、`create-configuration` AWS CLI コマンドを使用して新しい設定を作成します。

```
aws mq create-configuration \
  --name "rabbitmq-ldap-config" \
  --engine-type "RABBITMQ" \
  --engine-version "3.13"
```

このコマンドでは、次の例のようなレスポンスが返されます。

```
{
"Arn": "arn:aws:mq:us-west-2:123456789012:configuration:c-fa3390a5-7e01-4559-ae0c-
eb15b38b22ca",
```

```
    "AuthenticationStrategy": "simple",
    "Created": "2025-07-17T16:03:01.759943+00:00",
    "Id": "c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca",
    "LatestRevision": {
"Created": "2025-07-17T16:03:01.759000+00:00",
    "Description": "Auto-generated default for rabbitmq-ldap-config on RabbitMQ
 3.13",
    "Revision": 1
    },
    "Name": "rabbitmq-ldap-config"
}
```

2. 次の例に示すように、LDAP rabbitmq.confを認証および認可方法として使用する という設定ファイルを作成します。テンプレート内のすべてのプレースホルダー値 (でマーク) を${RabbitMqLdapTestStack.*}、デプロイされた AWS CDK 前提条件スタック出力または同等のインフラストラクチャの実際の値に置き換えます。

```
auth_backends.1 = ldap

# LDAP authentication settings - For more information,
# see https://www.rabbitmq.com/docs/ldap#basic

# FIXME: Replace the ${RabbitMqLdapTestStack.*} placeholders with actual values
# from your deployed prerequisite CDK stack outputs.
auth_ldap.servers.1 = ${RabbitMqLdapTestStack.NlbDnsName}
auth_ldap.dn_lookup_bind.user_dn = ${RabbitMqLdapTestStack.DnLookupUserDn}
auth_ldap.dn_lookup_base = ${RabbitMqLdapTestStack.DnLookupBase}
auth_ldap.dn_lookup_attribute = ${RabbitMqLdapTestStack.DnLookupAttribute}
auth_ldap.port = 636
auth_ldap.use_ssl = true
auth_ldap.ssl_options.verify = verify_peer
auth_ldap.log = network

# AWS integration for secure credential retrieval
# - see: https://github.com/amazon-mq/rabbitmq-aws
# The aws plugin allows RabbitMQ to securely retrieve credentials and certificates
# from AWS services.

# Replace the ${RabbitMqLdapTestStack.*} placeholders with actual ARN values
# from your deployed prerequisite CDK stack outputs.
aws.arns.auth_ldap.ssl_options.cacertfile = ${RabbitMqLdapTestStack.CaCertArn}
```

```
aws.arns.auth_ldap.dn_lookup_bind.password =
 ${RabbitMqLdapTestStack.DnLookupUserPasswordArn}
aws.arns.assume_role_arn = ${RabbitMqLdapTestStack.AmazonMqAssumeRoleArn}

# LDAP authorization queries - For more information,
# see: https://www.rabbitmq.com/docs/ldap#authorisation

# FIXME: Replace the ${RabbitMqLdapTestStack.*} placeholders with actual group DN
# values from your deployed prerequisite CDK stack outputs
# Uses Active Directory groups created by the prerequisite CDK stack
auth_ldap.queries.tags = '''
[{administrator, {in_group,
 "${RabbitMqLdapTestStack.RabbitMqAdministratorsGroupDn}"}},
{management,    {in_group,
 "${RabbitMqLdapTestStack.RabbitMqMonitoringUsersGroupDn}"}}]
'''

# FIXME: This provides all authenticated users access to all vhosts
# - update to restrict access as required
auth_ldap.queries.vhost_access = '''
{constant, true}
'''

# FIXME: This provides all authenticated users full access to all
# queues and exchanges - update to restrict access as required
auth_ldap.queries.resource_access = '''
{for, [    {permission, configure, {constant, true}},
     {permission, write,
      {for, [{resource, queue,    {constant, true}},
             {resource, exchange, {constant, true}}]}},
     {permission, read,
      {for, [{resource, exchange, {constant, true}},
             {resource, queue,    {constant, true}}]}}
    ]
}
'''

# FIXME: This provides all authenticated users access to all topics
# - update to restrict access as required
auth_ldap.queries.topic_access = '''
{for, [{permission, write, {constant, true}},
     {permission, read,  {constant, true}}
    ]
}
```

```
'''
```

3. 次の例に示すように、update-configuration AWS CLI コマンドを使用して設定を更新します。このコマンドでは、この手順のステップ 1 のレスポンスで受け取った設定 ID を追加します。例えば、c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca。

```
aws mq update-configuration \
  --configuration-id "<c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca>" \
  --data "$(cat rabbitmq.conf | base64 --wrap=0)"
```

このコマンドでは、次の例のようなレスポンスが返されます。

```
{
    "Arn": "arn:aws:mq:us-west-2:123456789012:configuration:c-b600ac8e-8183-4f74-
a713-983e59f30e3d",
    "Created": "2025-07-17T16:57:04.520931+00:00",
    "Id": "c-b600ac8e-8183-4f74-a713-983e59f30e3d",
    "LatestRevision": {
        "Created": "2025-07-17T16:57:39.172000+00:00",
        "Revision": 2
    },
    "Name": "rabbitmq-ldap-config",
    "Warnings": []
}
```

4. この手順のステップ 2 で作成した LDAP 設定を使用してブローカーを作成します。これを行うには、次の例に示すように create-broker AWS CLI コマンドを使用します。このコマンドでは、ステップ 1 と 2 のレスポンスで取得した設定 ID とリビジョン番号をそれぞれ指定します。例えば、c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca と 2 です。

```
aws mq create-broker \
 --broker-name "rabbitmq-ldap-test-1" \
 --engine-type "RABBITMQ" \
 --engine-version "3.13" \
 --host-instance-type "mq.m7g.large" \
 --deployment-mode "CLUSTER_MULTI_AZ" \
```

```
 --logs '{"General": true}' \
 --publicly-accessible \
 --configuration '{"Id": "<c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca>","Revision":
 <2>}'
```

このコマンドでは、次の例のようなレスポンスが返されます。

```
{
    "BrokerArn": "arn:aws:mq:us-west-2:123456789012:broker:rabbitmq-ldap-
broker:b-2a1b5133-a10c-49d2-879b-8c176c34cf73",
    "BrokerId": "b-2a1b5133-a10c-49d2-879b-8c176c34cf73"
}
```

> **⚠ ブローカーの命名制限**
>
> 前提条件の CDK スタックによって作成された IAM ロールは、ブローカー名を で始まるように制限しますrabbitmq-ldap-test。ブローカー名がこのパターンに従っているか、IAM ロールに ARN 解決のためにロールを引き受けるアクセス許可がないことを確認します。

5. 次の例に示すように describe-broker AWS CLI コマンドを使用してRUNNING、ブローカーのステータスが から CREATION_IN_PROGRESSに移行していることを確認します。このコマンドでは、前のステップの結果で取得したブローカー ID を指定します (例: b-2a1b5133-a10c-49d2-879b-8c176c34cf73)。

```
aws mq describe-broker \
 --broker-id "<b-2a1b5133-a10c-49d2-879b-8c176c34cf73>"
```

このコマンドでは、次の例のようなレスポンスが返されます。次のレスポンスは、describe-broker コマンドが返す完全な出力の省略バージョンです。このレスポンスは、ブローカーのステータスと、ブローカーの保護に使用される認証戦略を示します。この場合、config_managed認証戦略はブローカーが LDAP 認証方法を使用していることを示します。

```
{
"AuthenticationStrategy": "config_managed",
    ...,
    "BrokerState": "RUNNING",
    ...
}
```

6. 前提条件の CDK スタックによって作成されたテストユーザーの 1 人を使用して RabbitMQ アクセスを検証する

```
# FIXME: Replace ${RabbitMqLdapTestStack.ConsoleUserPasswordArn} with the actual
 ARN from your deployed prerequisite CDK stack outputs
CONSOLE_PASSWORD=$(aws secretsmanager get-secret-value \
  --secret-id ${RabbitMqLdapTestStack.ConsoleUserPasswordArn} \
  --query 'SecretString' --output text)

# FIXME: Replace BrokerConsoleURL with the actual ConsoleURL retrieved by
# calling describe-broker for the broker created above
# Call management API /api/overview (should succeed)
curl -u RabbitMqConsoleUser:$CONSOLE_PASSWORD \
  https://${BrokerConsoleURL}/api/overview

# Try to create a user (should fail - console user only has monitoring permissions)
curl -u RabbitMqConsoleUser:$CONSOLE_PASSWORD \
  -X PUT https://${BrokerConsoleURL}/api/users/testuser \
  -H "Content-Type: application/json" \
  -d '{"password":"testpass","tags":"management"}'
```

Amazon MQ for RabbitMQ での HTTP 認証と認可の使用

このチュートリアルでは、外部 HTTP サーバーを使用して Amazon MQ for RabbitMQ ブローカーの HTTP 認証と認可を設定する方法について説明します。

> **Note**
>
> HTTP 認証プラグインは、Amazon MQ for RabbitMQ バージョン 4 以降でのみ使用できます。

このページの内容

HTTP 認証と認可を設定するための前提条件

このチュートリアルで必要な AWS リソースを設定するには、AWS Amazon MQ for RabbitMQ HTTP 認証統合用の CDK スタックをデプロイします。

この CDK スタックは、HTTP 認証サーバー、証明書、IAM ロールなど、必要なすべての AWS リソースを自動的に作成します。スタックによって作成されたリソースの完全なリストについては、パッケージ README を参照してください。

CDK スタックを使用する代わりにリソースを手動で設定する場合は、Amazon MQ for RabbitMQ ブローカーで HTTP 認証を設定する前に、同等のインフラストラクチャが整っていることを確認してください。

Amazon MQ をセットアップするための前提条件

AWS CLI バージョン >= 2.28.23。ブローカーの作成時にユーザー名とパスワードをオプションで追加します。

CLI を使用した RabbitMQ での HTTP AWS 認証の設定

この手順では、 CLI AWS を使用して必要なリソースを作成および設定します。次の手順では、プレースホルダー値を実際の値に置き換えてください。

1. 次の例に示すように、`create-configuration` AWS CLI コマンドを使用して新しい設定を作成します。

```
aws mq create-configuration \
  --name "rabbitmq-http-config" \
```

```
  --engine-type "RABBITMQ" \
  --engine-version "4.2"
```

このコマンドでは、次の例のようなレスポンスが返されます。

```
{
    "Arn": "arn:aws:mq:us-west-2:123456789012:configuration:c-fa3390a5-7e01-4559-
ae0c-eb15b38b22ca",
    "AuthenticationStrategy": "simple",
    "Created": "2025-07-17T16:03:01.759943+00:00",
    "Id": "c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca",
    "LatestRevision": {
        "Created": "2025-07-17T16:03:01.759000+00:00",
        "Description": "Auto-generated default for rabbitmq-http-config on RabbitMQ
 4.2",
        "Revision": 1
    },
    "Name": "rabbitmq-http-config"
}
```

2. 次の例に示すように、HTTP `rabbitmq.conf`を認証および認可方法として使用する という設定ファイルを作成します。テンプレート内のすべてのプレースホルダー値 (でマーク) を`${...}`、デプロイされた AWS CDK 前提条件スタック出力または同等のインフラストラクチャの実際の値に置き換えます。

```
auth_backends.1 = cache
auth_backends.2 = http
auth_cache.cached_backend = http

# HTTP authentication settings
# For more information, see https://github.com/rabbitmq/rabbitmq-auth-backend-http

# FIXME: Replace the ${...} placeholders with actual values
# from your deployed prerequisite CDK stack outputs.
auth_http.http_method = post
auth_http.user_path = ${HttpServerUserPath}
auth_http.vhost_path = ${HttpServerVhostPath}
auth_http.resource_path = ${HttpServerResourcePath}
```

```
auth_http.topic_path = ${HttpServerTopicPath}

# TLS/HTTPS configuration
auth_http.ssl_options.verify = verify_peer
auth_http.ssl_options.sni = test.amazonaws.com

# AWS integration for secure credential retrieval
# For more information, see https://github.com/amazon-mq/rabbitmq-aws

# Replace the ${...} placeholders with actual ARN values
# from your deployed prerequisite CDK stack outputs.
aws.arns.assume_role_arn = ${AmazonMqAssumeRoleArn}
aws.arns.auth_http.ssl_options.cacertfile = ${CaCertArn}
```

3. update-configuration AWS CLI コマンドを使用して設定を更新します。ステップ 3 の設定 ID を使用します。

```
aws mq update-configuration \
  --configuration-id "<c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca>" \
  --data "$(cat rabbitmq.conf | base64 --wrap=0)"
```

このコマンドでは、次の例のようなレスポンスが返されます。

```
{
    "Arn": "arn:aws:mq:us-west-2:123456789012:configuration:c-fa3390a5-7e01-4559-
ae0c-eb15b38b22ca",
    "Created": "2025-07-17T16:57:04.520931+00:00",
    "Id": "c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca",
    "LatestRevision": {
        "Created": "2025-07-17T16:57:39.172000+00:00",
        "Revision": 2
    },
    "Name": "rabbitmq-http-config",
    "Warnings": []
}
```

4. HTTP 設定でブローカーを作成します。前のステップの設定 ID とリビジョン番号を使用します。

```
aws mq create-broker \
  --broker-name "rabbitmq-http-test-1" \
  --engine-type "RABBITMQ" \
  --engine-version "4.2" \
  --host-instance-type "mq.m7g.large" \
  --deployment-mode "SINGLE_INSTANCE" \
  --logs '{"General": true}' \
  --publicly-accessible \
  --configuration '{"Id": "<c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca>","Revision":
 <2>}'
```

このコマンドでは、次の例のようなレスポンスが返されます。

```
{
    "BrokerArn": "arn:aws:mq:us-west-2:123456789012:broker:rabbitmq-http-
test-1:b-2a1b5133-a10c-49d2-879b-8c176c34cf73",
    "BrokerId": "b-2a1b5133-a10c-49d2-879b-8c176c34cf73"
}
```

5. describe-broker AWS CLI コマンドを使用してRUNNING、ブローカーのステータスが からCREATION_IN_PROGRESSに移行していることを確認します。

```
aws mq describe-broker \
  --broker-id "<b-2a1b5133-a10c-49d2-879b-8c176c34cf73>"
```

このコマンドでは、次の例のようなレスポンスが返されます。config_managed 認証戦略は、ブローカーが HTTP 認証方法を使用していることを示します。

```
{
    "AuthenticationStrategy": "config_managed",
    ...,
```

```
    "BrokerState": "RUNNING",
    ...
}
```

6. 前提条件の CDK スタックによって作成されたテストユーザーの 1 人を使用して RabbitMQ アクセスを検証する

```
# FIXME: Replace ${RabbitMqHttpAuthElbStack.ConsoleUserPasswordArn} with the actual
 ARN from your deployed prerequisite CDK stack outputs
CONSOLE_PASSWORD=$(aws secretsmanager get-secret-value \
  --secret-id ${RabbitMqHttpAuthElbStack.ConsoleUserPasswordArn} \
  --query 'SecretString' --output text)

# FIXME: Replace BrokerConsoleURL with the actual ConsoleURL retrieved by
# calling describe-broker for the broker created above
# Call management API /api/overview (should succeed)
curl -u RabbitMqConsoleUser:$CONSOLE_PASSWORD \
  https://${BrokerConsoleURL}/api/overview

# Try to create a vhost (should fail - console user only has management
 permissions)
curl -u RabbitMqConsoleUser:$CONSOLE_PASSWORD \
  -X PUT https://${BrokerConsoleURL}/api/vhosts/test-vhost \
  -H "Content-Type: application/json" \
  -d '{}'
```

Amazon MQ for RabbitMQ での SSL 証明書認証の使用

このチュートリアルでは、プライベート認証機関を使用して Amazon MQ for RabbitMQ ブローカーの SSL 証明書認証を設定する方法について説明します。

> **Note**
>
> SSL 証明書認証プラグインは、Amazon MQ for RabbitMQ バージョン 4 以降でのみ使用できます。

このページの内容

- SSL 証明書認証を設定するための前提条件
- CLI を使用した RabbitMQ での SSL AWS 証明書認証の設定

SSL 証明書認証を設定するための前提条件

SSL 証明書認証は、相互 TLS (mTLS) を使用して X.509 証明書を使用してクライアントを認証します。このチュートリアルで必要な AWS リソースを設定するには、AWS Amazon MQ for RabbitMQ mTLS 統合用の CDK スタックをデプロイします。

この CDK スタックは、認証機関、クライアント証明書、IAM ロールなど、必要なすべての AWS リソースを自動的に作成します。スタックによって作成されたリソースの完全なリストについては、パッケージ README を参照してください。

> **Note**
>
> CDK スタックをデプロイする前に、 `RABBITMQ_TEST_USER_NAME`環境変数を設定します。この値はクライアント証明書の共通名 (CN) として使用され、チュートリアルのステップで使用するユーザー名と一致する必要があります。例: `export RABBITMQ_TEST_USER_NAME="myuser"`

CDK スタックを使用する代わりにリソースを手動で設定する場合は、Amazon MQ for RabbitMQ ブローカーで SSL 証明書認証を設定する前に、同等のインフラストラクチャが整っていることを確認してください。

Amazon MQ をセットアップするための前提条件

AWS CLI バージョン >= 2.28.23。ブローカーの作成時にユーザー名とパスワードをオプションで追加します。

CLI を使用した RabbitMQ での SSL AWS 証明書認証の設定

この手順では、 CLI AWS を使用して必要なリソースを作成および設定します。次の手順では、configurationID や Revision、 `<c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca>` などのプレースホルダー値を実際の値に置き換え<2>てください。

1. 次の例に示すように、`create-configuration` AWS CLI コマンドを使用して新しい設定を作成します。

```
aws mq create-configuration \
  --name "rabbitmq-ssl-config" \
  --engine-type "RABBITMQ" \
  --engine-version "4.2"
```

このコマンドでは、次の例のようなレスポンスが返されます。

```
{
    "Arn": "arn:aws:mq:us-west-2:123456789012:configuration:c-fa3390a5-7e01-4559-
ae0c-eb15b38b22ca",
    "AuthenticationStrategy": "simple",
    "Created": "2025-07-17T16:03:01.759943+00:00",
    "Id": "c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca",
    "LatestRevision": {
        "Created": "2025-07-17T16:03:01.759000+00:00",
        "Description": "Auto-generated default for rabbitmq-ssl-config on RabbitMQ
 4.2",
        "Revision": 1
    },
    "Name": "rabbitmq-ssl-config"
}
```

2. 次の例に示すように、SSL 証明書認証を使用するrabbitmq.confように という設定ファイルを作成します。テンプレート内のすべてのプレースホルダー値 (でマーク) を${...}、デプロイされた AWS CDK 前提条件スタック出力または同等のインフラストラクチャの実際の値に置き換えます。

```
auth_mechanisms.1 = EXTERNAL
ssl_cert_login_from = common_name

auth_backends.1 = internal

# Reject if no client cert
ssl_options.verify = verify_peer
ssl_options.fail_if_no_peer_cert = true
```

```
# AWS integration for secure credential retrieval
# For more information, see https://github.com/amazon-mq/rabbitmq-aws

# FIXME: Replace the ${...} placeholders with actual ARN values
# from your deployed prerequisite CDK stack outputs.
aws.arns.assume_role_arn = ${AmazonMqAssumeRoleArn}
aws.arns.ssl_options.cacertfile = ${CaCertArn}
```

3. 次の例に示すように、update-configuration AWS CLI コマンドを使用して設定を更新します。このコマンドでは、この手順のステップ 1 のレスポンスで受け取った設定 ID を追加します。例えば、c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca。

```
aws mq update-configuration \
  --configuration-id "<c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca>" \
  --data "$(cat rabbitmq.conf | base64 --wrap=0)"
```

このコマンドでは、次の例のようなレスポンスが返されます。

```
{
    "Arn": "arn:aws:mq:us-west-2:123456789012:configuration:c-fa3390a5-7e01-4559-
ae0c-eb15b38b22ca",
    "Created": "2025-07-17T16:57:04.520931+00:00",
    "Id": "c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca",
    "LatestRevision": {
        "Created": "2025-07-17T16:57:39.172000+00:00",
        "Revision": 2
    },
    "Name": "rabbitmq-ssl-config",
    "Warnings": []
}
```

4. この手順のステップ 2 で作成した SSL 証明書認証設定を使用してブローカーを作成します。これを行うには、次の例に示すように create-broker AWS CLI コマンドを使用します。このコマンドでは、ステップ 1 と 2 のレスポンスで取得した設定 ID とリビジョン番号をそれぞれ指定します。例えば、c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca と 2 です。

```
aws mq create-broker \
  --broker-name "rabbitmq-ssl-test-1" \
  --engine-type "RABBITMQ" \
  --engine-version "4.2" \
  --host-instance-type "mq.m7g.large" \
  --deployment-mode "SINGLE_INSTANCE" \
  --logs '{"General": true}' \
  --publicly-accessible \
  --configuration '{"Id": "<c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca>","Revision":
 <2>}' \
  --users '[{"Username":"testuser","Password":"testpassword"}]'
```

このコマンドでは、次の例のようなレスポンスが返されます。

```
{
    "BrokerArn": "arn:aws:mq:us-west-2:123456789012:broker:rabbitmq-ssl-
test-1:b-2a1b5133-a10c-49d2-879b-8c176c34cf73",
    "BrokerId": "b-2a1b5133-a10c-49d2-879b-8c176c34cf73"
}
```

5. 次の例に示すように describe-broker AWS CLI コマンドを使用してRUNNING、ブローカーのステータスが から CREATION_IN_PROGRESSに移行していることを確認します。このコマンドでは、前のステップの結果で取得したブローカー ID を指定します。例えば、b-2a1b5133-a10c-49d2-879b-8c176c34cf73。

```
aws mq describe-broker \
  --broker-id "<b-2a1b5133-a10c-49d2-879b-8c176c34cf73>"
```

このコマンドでは、次の例のようなレスポンスが返されます。次のレスポンスは、describe-broker コマンドが返す完全な出力の省略バージョンです。このレスポンスは、ブローカーのステータスと、ブローカーの保護に使用される認証戦略を示します。この場合、config_managed認証戦略はブローカーが SSL 証明書認証方法を使用していることを示します。

```
{
```

```
    "AuthenticationStrategy": "config_managed",
    ...,
    "BrokerState": "RUNNING",
    ...
}
```

6. 次のssl.shスクリプトを使用して SSL 証明書認証を検証します。

   この bash スクリプトを使用して、Amazon MQ for RabbitMQ ブローカーへの接続をテストします。このスクリプトは、認証にクライアント証明書を使用し、接続が正しく設定されているかどうかを確認します。正常に設定されると、ブローカーがメッセージを発行して消費します。

   ACCESS_REFUSED エラーが発生した場合は、ブローカーの CloudWatch ログを使用して設定をトラブルシューティングできます。ブローカーの CloudWatch ロググループのリンクは、Amazon MQ コンソールにあります。

   このスクリプトでは、次の値を指定する必要があります。

   - USERNAME: クライアント証明書の共通名 (CN)。
   - CLIENT_KEYSTORE: クライアントキーストアファイルへのパス (PKCS12 形式) 。前提条件の CDK スタックを使用した場合、デフォルトのパスは です$(pwd)/certs/client-keystore.p12。
   - KEYSTORE_PASSWORD: クライアントキーストアのパスワード。前提条件の CDK スタックを使用した場合、デフォルトのパスワードは ですchangeit。
   - BROKER_DNS: この値は、Amazon MQ コンソールのブローカーの詳細ページの Connections にあります。

```
#! /bin/bash
set -e

# Client information
## FIXME: Update this value with the client ID and secret of your confidential
 application client
USERNAME=<client_cert_common_name>
CLIENT_KEYSTORE=$(pwd)/certs/client-keystore.p12
KEYSTORE_PASSWORD=changeit

BROKER_DNS=<broker_dns>
CONNECTION_STRING=amqps://${BROKER_DNS}:5671
```

```
# Produce/consume messages using the above connection string
QUEUES_COUNT=1
PRODUCERS_COUNT=1
CONSUMERS_COUNT=1
PRODUCER_RATE=1

finch run --rm --ulimit nofile=40960:40960 \
    -v ${CLIENT_KEYSTORE}:/certs/client-keystore.p12:ro \
    -e JAVA_TOOL_OPTIONS="-Djavax.net.ssl.keyStore=/certs/client-
keystore.p12 -Djavax.net.ssl.keyStorePassword=${KEYSTORE_PASSWORD} -
Djavax.net.ssl.keyStoreType=PKCS12" \
    pivotalrabbitmq/perf-test:latest \
    --queue-pattern 'test-queue-cert-%d' --queue-pattern-from 1 --queue-pattern-to
 $QUEUES_COUNT \
    --producers $PRODUCERS_COUNT --consumers $CONSUMERS_COUNT \
    --id "cert-test${QUEUES_COUNT}q${PRODUCERS_COUNT}p${CONSUMERS_COUNT}c
${PRODUCER_RATE}r" \
    --uri ${CONNECTION_STRING} \
    --sasl-external \
    --use-default-ssl-context \
    --flag persistent --rate $PRODUCER_RATE
```

AMQP および管理エンドポイントに mTLS を使用する

このチュートリアルでは、プライベート認証機関を使用して AMQP クライアント接続と RabbitMQ 管理インターフェイスの相互 TLS (mTLS) を設定する方法について説明します。

> **Note**
>
> mTLS のプライベート認証機関の使用は、Amazon MQ for RabbitMQ バージョン 4 以降でのみ使用できます。

このページの内容

mTLS を設定するための前提条件

このチュートリアルで必要な AWS リソースを設定するには、AWS Amazon MQ for RabbitMQ mTLS 統合用の CDK スタックを にデプロイします。

この CDK スタックは、認証機関、クライアント証明書、IAM ロールなど、必要なすべての AWS リソースを自動的に作成します。スタックによって作成されたリソースの完全なリストについては、パッケージ README を参照してください。

CDK スタックを使用する代わりにリソースを手動で設定する場合は、Amazon MQ for RabbitMQ ブローカーで mTLS を設定する前に、同等のインフラストラクチャが整っていることを確認してください。

Amazon MQ をセットアップするための前提条件

AWS CLI バージョン >= 2.28.23。ブローカーの作成時にユーザー名とパスワードをオプションで追加します。

CLI を使用した RabbitMQ での mTLS AWS の設定

この手順では、 CLI AWS を使用して必要なリソースを作成および設定します。次の手順では、configurationID や Revision、 <c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca> などのプレースホルダー値を実際の値に置き換え<2>てください。

1. 次の例に示すように、create-configuration AWS CLI コマンドを使用して新しい設定を作成します。

```
aws mq create-configuration \
  --name "rabbitmq-mtls-config" \
  --engine-type "RABBITMQ" \
  --engine-version "4.2"
```

このコマンドでは、次の例のようなレスポンスが返されます。

```
{
    "Arn": "arn:aws:mq:us-west-2:123456789012:configuration:c-fa3390a5-7e01-4559-
ae0c-eb15b38b22ca",
    "AuthenticationStrategy": "simple",
```

```
    "Created": "2025-07-17T16:03:01.759943+00:00",
    "Id": "c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca",
    "LatestRevision": {
        "Created": "2025-07-17T16:03:01.759000+00:00",
        "Description": "Auto-generated default for rabbitmq-mtls-config on RabbitMQ
 4.2",
        "Revision": 1
    },
    "Name": "rabbitmq-mtls-config"
}
```

2. 次の例に示すように、 という設定ファイルを作成して、AMQP および管理エンドポイントの mTLS rabbitmq.confを設定します。テンプレート内のすべてのプレースホルダー値 (でマーク) を${...}、デプロイされた AWS CDK 前提条件スタック出力または同等のインフラストラクチャの実際の値に置き換えます。

```
auth_backends.1 = internal

# TLS configuration
ssl_options.verify = verify_peer
ssl_options.fail_if_no_peer_cert = true
management.ssl.verify = verify_peer

# AWS integration for secure credential retrieval
# For more information, see https://github.com/amazon-mq/rabbitmq-aws

# FIXME: Replace the ${...} placeholders with actual ARN values
# from your deployed prerequisite CDK stack outputs.
aws.arns.assume_role_arn = ${AmazonMqAssumeRoleArn}
aws.arns.ssl_options.cacertfile = ${CaCertArn}
aws.arns.management.ssl.cacertfile = ${CaCertArn}
```

3. 次の例に示すように、update-configuration AWS CLI コマンドを使用して設定を更新します。このコマンドでは、この手順のステップ 1 のレスポンスで受け取った設定 ID を追加します。例えば、c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca。

```
aws mq update-configuration \
  --configuration-id "<c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca>" \
```

```
  --data "$(cat rabbitmq.conf | base64 --wrap=0)"
```

このコマンドでは、次の例のようなレスポンスが返されます。

```
{
    "Arn": "arn:aws:mq:us-west-2:123456789012:configuration:c-fa3390a5-7e01-4559-
ae0c-eb15b38b22ca",
    "Created": "2025-07-17T16:57:04.520931+00:00",
    "Id": "c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca",
    "LatestRevision": {
        "Created": "2025-07-17T16:57:39.172000+00:00",
        "Revision": 2
    },
    "Name": "rabbitmq-mtls-config",
    "Warnings": []
}
```

4. この手順のステップ 2 で作成した mTLS 設定を使用してブローカーを作成します。これを行うには、次の例に示すように create-broker AWS CLI コマンドを使用します。このコマンドでは、ステップ 1 と 2 のレスポンスで取得した設定 ID とリビジョン番号をそれぞれ指定します。例えば、c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca と 2 です。

```
aws mq create-broker \
  --broker-name "rabbitmq-mtls-test-1" \
  --engine-type "RABBITMQ" \
  --engine-version "4.2" \
  --host-instance-type "mq.m7g.large" \
  --deployment-mode "SINGLE_INSTANCE" \
  --logs '{"General": true}' \
  --publicly-accessible \
  --configuration '{"Id": "<c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca>","Revision":
 <2>}' \
  --users '[{"Username":"testuser","Password":"testpassword"}]'
```

このコマンドでは、次の例のようなレスポンスが返されます。

```
{
    "BrokerArn": "arn:aws:mq:us-west-2:123456789012:broker:rabbitmq-mtls-
test-1:b-2a1b5133-a10c-49d2-879b-8c176c34cf73",
    "BrokerId": "b-2a1b5133-a10c-49d2-879b-8c176c34cf73"
}
```

5. 次の例に示すように describe-broker AWS CLI コマンドを使用してRUNNING、ブローカーのステータスが から CREATION_IN_PROGRESSに移行していることを確認します。このコマンドでは、前のステップの結果で取得したブローカー ID を指定します。例えば、b-2a1b5133-a10c-49d2-879b-8c176c34cf73。

```
aws mq describe-broker \
  --broker-id "<b-2a1b5133-a10c-49d2-879b-8c176c34cf73>"
```

このコマンドでは、次の例のようなレスポンスが返されます。次のレスポンスは、describe-broker コマンドが返す完全な出力の省略バージョンです。

```
{
    "AuthenticationStrategy": "simple",
    ...,
    "BrokerState": "RUNNING",
    ...
}
```

6. 次のmtls.shスクリプトを使用して mTLS 認証を検証します。

この bash スクリプトを使用して、Amazon MQ for RabbitMQ ブローカーへの接続をテストします。このスクリプトは、クライアント証明書を使用して認証し、接続が正しく設定されているかどうかを確認します。正常に設定されると、ブローカーがメッセージを発行して消費します。

ACCESS_REFUSED エラーが発生した場合は、ブローカーの CloudWatch ログを使用して設定をトラブルシューティングできます。ブローカーの CloudWatch ロググループのリンクは、Amazon MQ コンソールにあります。

このスクリプトでは、次の値を指定する必要があります。

- USERNAME および PASSWORD: ブローカーで作成した RabbitMQ ユーザー認証情報。
- CLIENT_KEYSTORE: クライアントキーストアファイルへのパス (PKCS12 形式) 。前提条件の CDK スタックを使用した場合、デフォルトのパスは です$(pwd)/certs/client-keystore.p12。
- KEYSTORE_PASSWORD: クライアントキーストアのパスワード。前提条件の CDK スタックを使用した場合、デフォルトのパスワードは ですchangeit。
- BROKER_DNS: この値は、Amazon MQ コンソールのブローカーの詳細ページの Connections にあります。

```
#! /bin/bash
set -e

# Client information
## FIXME: Update this value with the client ID and secret of your confidential
 application client
USERNAME=<testuser>
PASSWORD=<testpassword>
CLIENT_KEYSTORE=$(pwd)/certs/client-keystore.p12
KEYSTORE_PASSWORD=changeit

BROKER_DNS=<broker_dns>
CONNECTION_STRING=amqps://${USERNAME}:${PASSWORD}@${BROKER_DNS}:5671

# Produce/consume messages using the above connection string
QUEUES_COUNT=1
PRODUCERS_COUNT=1
CONSUMERS_COUNT=1
PRODUCER_RATE=1

finch run --rm --ulimit nofile=40960:40960 \
    -v ${CLIENT_KEYSTORE}:/certs/client-keystore.p12:ro \
    -e JAVA_TOOL_OPTIONS="-Djavax.net.ssl.keyStore=/certs/client-
keystore.p12 -Djavax.net.ssl.keyStorePassword=${KEYSTORE_PASSWORD} -
Djavax.net.ssl.keyStoreType=PKCS12" \
    pivotalrabbitmq/perf-test:latest \
    --queue-pattern 'test-queue-cert-%d' --queue-pattern-from 1 --queue-pattern-to
 $QUEUES_COUNT \
    --producers $PRODUCERS_COUNT --consumers $CONSUMERS_COUNT \
    --id "cert-test${QUEUES_COUNT}q${PRODUCERS_COUNT}p${CONSUMERS_COUNT}c
${PRODUCER_RATE}r" \
```

```
    --uri ${CONNECTION_STRING} \
    --use-default-ssl-context \
    --flag persistent --rate $PRODUCER_RATE
```

JMS アプリケーションを接続する

このチュートリアルでは、RabbitMQ JMS クライアントを使用して JMS アプリケーションを Amazon MQ for RabbitMQ ブローカーに接続する方法を示します。 RabbitMQ プロデューサーを作成してメッセージを送信し、コンシューマーを作成して RabbitMQ キューからメッセージを受信する方法について説明します。

開始する前に、適切な RabbitMQ JMS 依存関係を Maven プロジェクトに追加します。

JMS 1.1 および 2.0 の場合:

```
<dependencies>

  <dependency>
    <groupId>com.rabbitmq.jms</groupId>
    <artifactId>rabbitmq-jms</artifactId>
    <version>2.12.0</version>
  </dependency>

</dependencies>
```

JMS 3.1 の場合:

```
<dependencies>

  <dependency>
    <groupId>com.rabbitmq.jms</groupId>
    <artifactId>rabbitmq-jms</artifactId>
    <version>3.5.0</version>
  </dependency>

</dependencies>
```

プロデューサーを作成する

次のコード例は、JMS を使用して RabbitMQ キューに書き込む方法を示しています。

```
import jakarta.jms.*;
import com.rabbitmq.jms.admin.*;

// Setting the connection factory
RMQConnectionFactory factory = new RMQConnectionFactory();
factory.setHost(envProps.getProperty("RABBITMQ_HOST", "localhost"));
factory.setPort(Integer.parseInt(envProps.getProperty("RABBITMQ_PORT", "5672")));
factory.setUsername(envProps.getProperty("RABBITMQ_USERNAME", "guest"));
factory.setPassword(envProps.getProperty("RABBITMQ_PASSWORD", "guest"));
factory.setVirtualHost(envProps.getProperty("RABBITMQ_VIRTUAL_HOST", "/"));
factory.useSslProtocol();

connection = factory.createConnection();
connection.start();

String queueName = "test-queue-jms";
Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);

RMQDestination destination = new RMQDestination(queueName, true, false);

// Send the message to the queue
MessageProducer producer = session.createProducer(destination);
producer.setDeliveryMode(DeliveryMode.PERSISTENT);

String msg_content = "Hello World!!";
TextMessage textMessage = session.createTextMessage(msg_content);
producer.send(textMessage);

System.out.printf("Published to AMQP queue '%s': %s", queueName, msg_content);
```

コンシューマーを作成する

次のコード例は、JMS を使用して RabbitMQ キューから読み取る方法を示しています。

```
import jakarta.jms.*;
import com.rabbitmq.jms.admin.*;

// Setting the connection factory
RMQConnectionFactory factory = new RMQConnectionFactory();
factory.setHost(envProps.getProperty("RABBITMQ_HOST", "localhost"));
factory.setPort(Integer.parseInt(envProps.getProperty("RABBITMQ_PORT", "5672")));
factory.setUsername(envProps.getProperty("RABBITMQ_USERNAME", "guest"));
factory.setPassword(envProps.getProperty("RABBITMQ_PASSWORD", "guest"));
```

```
factory.setVirtualHost(envProps.getProperty("RABBITMQ_VIRTUAL_HOST", "/"));
factory.useSslProtocol();

// Establish the connection and session
jakarta.jms.Connection connection = factory.createConnection();

String queueName = "test-queue-jms";
Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);

RMQDestination destination = new RMQDestination();
destination.setDestinationName(queueName);
destination.setAmqp(true);
destination.setAmqpQueueName(queueName);

// Initialize consumer
MessageConsumer consumer = session.createConsumer(destination);
consumer.setMessageListener(message -> {
    try {
        if (message instanceof TextMessage) {
            TextMessage textMessage = (TextMessage) message;
            System.out.printf("Message: %s%n", textMessage.getText());
        } else if (message instanceof BytesMessage) {
            BytesMessage bytesMessage = (BytesMessage) message;
            byte[] bytes = new byte[(int) bytesMessage.getBodyLength()];
            bytesMessage.readBytes(bytes);
            String content = new String(bytes);
            System.out.printf("Message: %s%n", content);
        } else {
            System.out.printf("Message: [%s]%n", message.getClass().getSimpleName());
        }
    } catch (JMSException e) {
        System.err.printf("Error processing message: %s%n", e.getMessage());
    }
});

connection.start();
```

Amazon MQ のセキュリティ

のクラウドセキュリティが最優先事項 AWS です。お客様は AWS 、セキュリティを最も重視する組織の要件を満たすように構築されたデータセンターとネットワークアーキテクチャからメリットを得られます。

セキュリティは、 AWS とお客様の間の責任共有です。責任共有モデルではこれをクラウドのセキュリティおよびクラウド内のセキュリティと説明しています。

- クラウドのセキュリティ – AWS クラウドで AWS サービスを実行するインフラストラクチャを保護する AWS 責任があります。 AWS また、 では、安全に使用できるサービスも提供しています。サードパーティーの監査者は、AWS コンプライアンスプログラムコンプライアンスプログラムの一環として、当社のセキュリティの有効性を定期的にテストおよび検証。Amazon MQ に適用されるコンプライアンスプログラムの詳細については、「コンプライアンスプログラムAWS による対象範囲内のサービスコンプライアンスプログラム」を参照してください。
- クラウドのセキュリティ – お客様の責任は、使用する AWS サービスによって決まります。また、ユーザーは、データの機密性、会社の要件、適用される法律や規制など、その他の要因についても責任を負います。

このドキュメントは、Amazon MQ の使用時に責任共有モデルがどのように適用されるかを理解するために役立ちます。以下のトピックでは、セキュリティおよびコンプライアンス上の目的を達成するように Amazon MQ を設定する方法について説明します。また、Amazon MQ リソースのモニタリングや保護に役立つ他の AWS サービスの使用方法についても説明します。

トピック

- Amazon MQ のデータ保護
- Amazon MQ のための Identity and Access Management
- Amazon MQ のコンプライアンス検証
- Amazon MQ の耐障害性
- Amazon MQ のインフラストラクチャセキュリティ
- Amazon MQ のセキュリティベストプラクティス

Amazon MQ のデータ保護

責任 AWS 共有モデル、Amazon MQ でのデータ保護に適用されます。このモデルで説明されているように、 AWS はすべての を実行するグローバルインフラストラクチャを保護する責任があります AWS クラウド。ユーザーは、このインフラストラクチャでホストされるコンテンツに対する管理を維持する責任があります。また、使用する「 AWS のサービス 」のセキュリティ設定と管理タスクもユーザーの責任となります。データプライバシーの詳細については、データプライバシーに関するよくある質問を参照してください。欧州でのデータ保護の詳細については、AWS セキュリティブログに投稿された「AWS 責任共有モデルおよび GDPR」のブログ記事を参照してください。

データ保護の目的で、認証情報を保護し AWS アカウント 、 AWS IAM アイデンティティセンターまたは AWS Identity and Access Management (IAM) を使用して個々のユーザーを設定することをお勧めします。この方法により、それぞれのジョブを遂行するために必要な権限のみが各ユーザーに付与されます。また、次の方法でデータを保護することもお勧めします:

- 各アカウントで多要素認証 (MFA) を使用します。
- SSL/TLS を使用して AWS リソースと通信します。TLS 1.2 は必須ですが、TLS 1.3 を推奨します。
- で API とユーザーアクティビティのログ記録を設定します AWS CloudTrail。CloudTrail 証跡を使用して AWS アクティビティをキャプチャする方法については、「 AWS CloudTrail ユーザーガイド」のCloudTrail 証跡の使用」を参照してください。
- AWS 暗号化ソリューションと、 内のすべてのデフォルトのセキュリティコントロールを使用します AWS のサービス。
- Amazon Macie などの高度な管理されたセキュリティサービスを使用します。これらは、Amazon S3 に保存されている機密データの検出と保護を支援します。
- コマンドラインインターフェイスまたは API AWS を介して にアクセスするときに FIPS 140-3 検証済みの暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、「連邦情報処理規格 (FIPS) 140-3」を参照してください。

お客様の E メールアドレスなどの極秘または機密情報を、タグ、または [名前] フィールドなどの自由形式のテキストフィールドに含めないことを強くお勧めします。これは、コンソール、API、または SDK を使用して Amazon MQ AWS CLIまたは他の AWS のサービス を使用する場合も同様です。AWS SDKs タグ、または名前に使用される自由記述のテキストフィールドに入力したデータは、請求または診断ログに使用される場合があります。外部サーバーに URL を提供する場合、そのサーバーへのリクエストを検証できるように、認証情報を URL に含めないことを強くお勧めします。

Amazon MQ for ActiveMQ と Amazon MQ for RabbitMQ ブローカーのどちらでも、ブローカーのウェブコンソールまたは Amazon MQ API を使用してリソースを作成するときに、ブローカー名またはユーザー名に個人を特定できる情報 (PII) またはその他の秘密情報や機密情報を使用しないでください。ブローカー名とユーザー名は、CloudWatch Logs を含む他の AWS のサービスからアクセスできます。ブローカーのユーザー名は、プライベートデータや機密データとして使用することを意図していません。

> **Important**
>
> TLS 1.3 は RabbitMQ ブローカーでは使用できません。

暗号化

Amazon MQ に保存されているユーザーデータは、保管中暗号化されています。Amazon MQ による保管時の暗号化は、 AWS Key Management Service (KMS) に保存されている暗号化キーを使用してデータを暗号化することによって、セキュリティを強化します。このサービスは、機密データの保護における負担と複雑な作業を減らすのに役立ちます。保管時に暗号化することで、セキュリティを重視したアプリケーションを構築して、暗号化のコンプライアンスと規制の要件を満たすことができます。

Amazon MQ ブローカー間のすべての接続は、転送時の暗号化を提供するために Transport layer Security (TLS) を使用します。

Amazon MQ は、Amazon MQ がセキュアな方法で管理して保存する暗号化キーを使用して、保管中および転送中のメッセージを暗号化します。詳細については、AWS Encryption SDK デベロッパーガイドを参照してください。

保管中の暗号化

Amazon MQ は AWS Key Management Service (KMS) と統合して、透過的なサーバー側の暗号化を提供します。Amazon MQ は、保管中のデータを常に暗号化します。

Amazon MQ for ActiveMQ ブローカーまたは Amazon MQ for RabbitMQ ブローカーを作成するときに、Amazon MQ AWS KMS key が保管中のデータの暗号化に使用する を指定できます。KMS キーを指定しない場合、Amazon MQ は AWS 所有の KMS キーを作成し、ユーザーに代わって使用します。Amazon MQ は現在、対称 KMS キーをサポートしています。KMS キーに関する詳細については、「AWS KMS keys」を参照してください。

ブローカーを作成するときに以下のいずれかを選択することによって、Amazon MQ が暗号化キーに何を使用するかを選択できます。

- Amazon MQ 所有の KMS キー (デフォルト) – キーは Amazon MQ が所有、管理し、ユーザーのアカウントにはありません。
- AWS マネージド KMS キー — AWS マネージド KMS キー (aws/mq) は、Amazon MQ によってユーザーに代わって作成、管理、使用されるアカウントの KMS キーです。
- 既存のカスタマーマネージド KMS キーを選択する – カスタマーマネージド KMS キーは、ユーザーが AWS Key Management Service (KMS) で作成し、管理します。

> **Important**
>
> - 付与の取り消しを元に戻すことはできません。ブローカーを削除して、アクセス権を取り消します。
> - Amazon Elastic File System (EFS) を使用してメッセージデータを保存する Amazon MQ for ActiveMQ ブローカーの場合、必要なアクションを実行した後にアカウント内の KMS キーを使用するアクセス許可が取り消されるまでに数時間かかることがあります。
> - EBS を使用してメッセージデータを保存する Amazon MQ for RabbitMQ ブローカーおよび Amazon MQ for ActiveMQ ブローカー場合、アカウントで KMS キーを使用する許可を Amazon EBS に提供する付与を無効にした、削除をスケジュールした、または取り消した場合、Amazon MQ はブローカーを維持できず、パフォーマンスが低下する可能性があります。
> - キーを無効にした場合、またはキーの削除をスケジュールした場合は、キーを再び有効にするか、キーの削除をキャンセルして、ブローカーの機能を維持できます。
> - 必要なアクションを実行した後、サーバーがキーを非アクティブ化したり、権限を取り消したりするには、数時間かかります。
> - CloudWatch ログを暗号化または復号するために、Amazon MQ が暗号化キーに使用するものを設定することはできません。CloudWatch のログは、暗号化を使用して保管中のデータを保護し、ロググループは暗号化されます。CloudWatch のログサービスは、デフォルトでサーバー側の暗号化を管理します。ロググループを暗号化する方法の詳細については、「Amazon CloudWatch Logs ユーザーガイド」を参照してください。

RabbitMQ の KMS キーを使用して単一インスタンスブローカーを作成すると、2 つの `CreateGrant` イベントが AWS CloudTrailに記録されます。最初のイベントは、Amazon MQ による KMS キー用の許可の作成です。2 つ目のイベントは、EBS による EBS 用の許可の作成です。

CreateGrant AWS CloudTrail ログエントリ: 単一インスタンスブローカー

mq_grant

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AKIAIOSFODNN7EXAMPLE",
        "arn": "arn:aws:iam::111122223333:user/AmazonMqConsole",
        "accountId": "111122223333",
        "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AKIAIOSFODNN7EXAMPLE",
                "arn": "arn:aws:iam::111122223333:user/AmazonMqConsole",
                "accountId": "111122223333",
                "userName": "AmazonMqConsole"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2023-02-23T18:59:10Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "mq.amazonaws.com"
    },
    "eventTime": "2018-06-28T22:23:46Z",
    "eventSource": "amazonmq.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "203.0.113.0",
    "userAgent": "PostmanRuntime/7.1.5",
    "requestParameters": {
        "granteePrincipal": "mq.amazonaws.com",
        "keyId": "arn:aws:kms:us-east-1:316438333700:key/bdbe42ae-f825-4e78-
a8a1-828d411c4be2",
```

```
        "retiringPrincipal": "mq.amazonaws.com",
        "operations": [
            "CreateGrant",
            "Decrypt",
            "GenerateDataKeyWithoutPlaintext",
            "ReEncryptFrom",
            "ReEncryptTo",
            "DescribeKey"
        ]
    },
    "responseElements": {
        "grantId":
 "0ab0ac0d0b000f00ea00cc0a0e00fc00bce000c000f0000000c0bc0a0000aaafSAMPLE",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",

    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": false,
    "resources": [
        {
           "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management",
    "sessionCredentialFromConsole": "true"
}
```

EBS grant creation

EBS の許可の作成に関するイベントが 1 つ表示されます。

```
                                {
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
```

```
        "invokedBy": "mq.amazonaws.com"
    },
    "eventTime": "2023-02-23T19:09:40Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "mq.amazonaws.com",
    "userAgent": "ExampleDesktop/1.0 (V1; OS)",
    "requestParameters": {
        "granteePrincipal": "mq.amazonaws.com",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
        "constraints": {
            "encryptionContextSubset": {
                "aws:ebs:id": "vol-0b670f00f7d5417c0"
            }
        },
        "operations": [
            "Decrypt"
        ],
        "retiringPrincipal": "ec2.us-east-1.amazonaws.com"
    },
    "responseElements": {
        "grantId":
 "0ab0ac0d0b000f00ea00cc0a0e00fc00bce000c000f0000000c0bc0a0000aaafSAMPLE",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
    },
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "sharedEventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventCategory": "Management"
```

```
}
```

RabbitMQ の KMS キーを使用してクラスターのデプロイを作成すると、5 つの CreateGrant イベントが AWS CloudTrailに記録されます。最初の 2 つのイベントは、Amazon MQ 用の許可の作成です。次の 3 つのイベントは、EBS による EBS 用の許可の作成です。

CreateGrant AWS CloudTrail ログエントリ: クラスターデプロイ

mq_grant

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AKIAIOSFODNN7EXAMPLE",
        "arn": "arn:aws:iam::111122223333:user/AmazonMqConsole",
        "accountId": "111122223333",
        "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AKIAIOSFODNN7EXAMPLE",
                "arn": "arn:aws:iam::111122223333:user/AmazonMqConsole",
                "accountId": "111122223333",
                "userName": "AmazonMqConsole"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2023-02-23T18:59:10Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "mq.amazonaws.com"
    },
    "eventTime": "2018-06-28T22:23:46Z",
    "eventSource": "amazonmq.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "203.0.113.0",
    "userAgent": "PostmanRuntime/7.1.5",
```

```
    "requestParameters": {
        "granteePrincipal": "mq.amazonaws.com",
        "keyId": "arn:aws:kms:us-east-1:316438333700:key/bdbe42ae-f825-4e78-
a8a1-828d411c4be2",
        "retiringPrincipal": "mq.amazonaws.com",
        "operations": [
            "CreateGrant",
            "Encrypt",
            "Decrypt",
            "ReEncryptFrom",
            "ReEncryptTo",
            "GenerateDataKey",
            "GenerateDataKeyWithoutPlaintext",
            "DescribeKey"
        ]
    },
    "responseElements": {
        "grantId":
 "0ab0ac0d0b000f00ea00cc0a0e00fc00bce000c000f0000000c0bc0a0000aaafSAMPLE",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",

    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": false,
    "resources": [
        {
           "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management",
    "sessionCredentialFromConsole": "true"
}
```

mq_rabbit_grant


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AKIAIOSFODNN7EXAMPLE",
        "arn": "arn:aws:iam::111122223333:user/AmazonMqConsole",
        "accountId": "111122223333",
        "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AKIAIOSFODNN7EXAMPLE",
                "arn": "arn:aws:iam::111122223333:user/AmazonMqConsole",
                "accountId": "111122223333",
                "userName": "AmazonMqConsole"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2023-02-23T18:59:10Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "mq.amazonaws.com"
    },
    "eventTime": "2018-06-28T22:23:46Z",
    "eventSource": "amazonmq.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "203.0.113.0",
    "userAgent": "PostmanRuntime/7.1.5",
    "requestParameters": {
        "granteePrincipal": "mq.amazonaws.com",
        "retiringPrincipal": "mq.amazonaws.com",
        "operations": [
            "DescribeKey"
        ],
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
    },
    "responseElements": {
```

```
        "grantId":
 "0ab0ac0d0b000f00ea00cc0a0e00fc00bce000c000f0000000c0bc0a0000aaafSAMPLE",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",

    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": false,
    "resources": [
        {
           "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management",
    "sessionCredentialFromConsole": "true"
}
```

EBS grant creation

EBS の許可の作成に関する 3 つのイベントが表示されます。

```
                                    {
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "mq.amazonaws.com"
    },
    "eventTime": "2023-02-23T19:09:40Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "mq.amazonaws.com",
    "userAgent": "ExampleDesktop/1.0 (V1; OS)",
    "requestParameters": {
        "granteePrincipal": "mq.amazonaws.com",
```

```
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
        "constraints": {
            "encryptionContextSubset": {
                "aws:ebs:id": "vol-0b670f00f7d5417c0"
            }
        },
        "operations": [
            "Decrypt"
        ],
        "retiringPrincipal": "ec2.us-east-1.amazonaws.com"
    },
    "responseElements": {
        "grantId":
 "0ab0ac0d0b000f00ea00cc0a0e00fc00bce000c000f0000000c0bc0a0000aaafSAMPLE",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
    },
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "sharedEventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventCategory": "Management"
}
```


KMS キーの詳細については、「AWS Key Management Service デベロッパーガイド」の「AWS KMS keys」を参照してください。

転送中の暗号化

Amazon MQ for ActiveMQ: Amazon MQ for ActiveMQ は強力な Transport Layer Security (TLS) を必要とし、Amazon MQ デプロイのブローカー間で転送されるデータを暗号化します。Amazon MQ ブローカー間で渡されるすべてのデータは、強力な Transport Layer Security (TLS) を使用して暗号化されています。これはすべての利用可能なプロトコルに当てはまります。

Amazon MQ for RabbitMQ: Amazon MQ for RabbitMQ は、すべてのクライアント接続に強力な Transport Layer Security (TLS) 暗号化を必要とします。RabbitMQ クラスターレプリケーショントラフィックはブローカーの VPC のみを転送し、 AWS データセンター間のすべてのネットワークトラフィックは物理レイヤーで透過的に暗号化されます。Amazon MQ for RabbitMQ クラスター化ブローカーは、現在、クラスターレプリケーションのノード間暗号化をサポートしていません。転送中のデータの詳細については、「保管中および転送中のデータの暗号化」を参照してください。

Amazon MQ for ActiveMQ のプロトコル

ActiveMQ ブローカーには、TLS が有効化されている以下のプロトコルを使用してアクセスできます。

- AMQP
- MQTT
- MQTT over WebSocket
- OpenWire
- STOMP
- STOMP over WebSocket

ActiveMQ 向けにサポートされている TLS 暗号スイート

ActiveMQ on Amazon MQ は、以下の暗号スイートをサポートしています。

- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
- TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA

- TLS_RSA_WITH_AES_256_GCM_SHA384
- TLS_RSA_WITH_AES_256_CBC_SHA256
- TLS_RSA_WITH_AES_256_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
- TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_AES_128_GCM_SHA256
- TLS_RSA_WITH_AES_128_CBC_SHA256
- TLS_RSA_WITH_AES_128_CBC_SHA

Amazon MQ for RabbitMQ のプロトコル

RabbitMQ ブローカーには、TLS が有効化されている以下のプロトコルを使用してアクセスできます。

- AMQP (0-9-1)

RabbitMQ 向けにサポートされている TLS 暗号スイート

RabbitMQ on Amazon MQ は、以下の暗号スイートをサポートしています。

- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

Amazon MQ のための Identity and Access Management

AWS Identity and Access Management (IAM) は、管理者が AWS リソースへのアクセスを安全に制御 AWS のサービス するのに役立つ です。IAM 管理者は、誰が認証 (サインイン) され、Amazon MQ リソースを使用する認可 を受ける (許可がある) ことができるかを制御します。IAM は、追加料金なしで使用できる AWS のサービス です。

トピック

オーディエンス

AWS Identity and Access Management (IAM) の使用方法は、ロールによって異なります。

- サービスユーザー - 機能にアクセスできない場合は、管理者にアクセス許可をリクエストします (「Amazon MQ アイデンティティとアクセスのトラブルシューティング」を参照)。
- サービス管理者 - ユーザーアクセスを決定し、アクセス許可リクエストを送信します (「Amazon MQ で IAM が機能する仕組み」を参照)
- IAM 管理者 - アクセスを管理するためのポリシーを作成します (「Amazon MQ のアイデンティティベースポリシーの例」を参照)

アイデンティティを使用した認証

認証とは、ID 認証情報 AWS を使用して にサインインする方法です。、IAM ユーザー AWS アカウントのルートユーザー、または IAM ロールを引き受けることで認証される必要があります。

AWS IAM アイデンティティセンター (IAM Identity Center)、シングルサインオン認証、Google/Facebook 認証情報などの ID ソースからの認証情報を使用して、フェデレーティッド ID としてサインインできます。サインインの詳細については、「AWS サインイン ユーザーガイド」の「AWS アカウントにサインインする方法」を参照してください。

プログラムによるアクセスの場合、 は SDK と CLI AWS を提供してリクエストを暗号化して署名します。詳細については、「IAM ユーザーガイド」の「API リクエストに対するAWS 署名バージョン 4」を参照してください。

AWS アカウント ルートユーザー

を作成するときは AWS アカウント、すべての AWS のサービス および リソースへの完全なアクセス権を持つ AWS アカウント ルートユーザーと呼ばれる 1 つのサインインアイデンティティから始めます。日常的なタスクには、ルートユーザーを使用しないことを強くお勧めします。ルートユーザー認証情報を必要とするタスクについては、「IAM ユーザーガイド」の「ルートユーザー認証情報が必要なタスク」を参照してください。

ユーザーとグループ

IAM ユーザーは、1 人のユーザーまたは 1 つのアプリケーションに対して特定のアクセス許可を持つ ID です。長期認証情報を持つ IAM ユーザーの代わりに一時的な認証情報を使用することをお勧めします。詳細については、IAM ユーザーガイドの「ID プロバイダーとのフェデレーションを使用してにアクセスする必要がある AWS」を参照してください。

IAM グループは、IAM ユーザーの集合を指定し、大量のユーザーに対するアクセス許可の管理を容易にします。詳細については、「IAM ユーザーガイド」の「IAM ユーザーに関するユースケース」を参照してください。

IAM ロール

IAM ロールは、特定のアクセス許可を持つアイデンティであり、一時的な認証情報を提供します。ユーザーから IAM ロール (コンソール) に切り替えるか、 または API オペレーションを呼び出すことで、ロールを引き受けることができます。 AWS CLI AWS 詳細については、「IAM ユーザーガイド」の「ロールを引き受けるための各種方法」を参照してください。

IAM ロールは、フェデレーションユーザーアクセス、一時的な IAM ユーザーのアクセス許可、クロスアカウントアクセス、クロスサービスアクセス、および Amazon EC2 で実行するアプリケーションに役立ちます。詳細については、IAM ユーザーガイド の IAM でのクロスアカウントリソースアクセス を参照してください。

ポリシーを使用したアクセスの管理

でアクセスを制御する AWS には、ポリシーを作成し、ID AWS またはリソースにアタッチします。ポリシーは、アイデンティティまたはリソースに関連付けられている場合のアクセス許可を定義します。 は、プリンシパルがリクエストを行うときにこれらのポリシー AWS を評価します。ほとんどのポリシーは JSON ドキュメント AWS として に保存されます。JSON ポリシードキュメントの詳細については、「IAM ユーザーガイド」の「JSON ポリシー概要」を参照してください。

管理者は、ポリシーを使用して、どのプリンシパルがどのリソースに対して、どのような条件でアクションを実行できるかを定義することで、誰が何にアクセスできるかを指定します。

デフォルトでは、ユーザーやロールにアクセス許可はありません。IAM 管理者は IAM ポリシーを作成してロールに追加し、このロールをユーザーが引き受けられるようにします。IAM ポリシーは、オペレーションの実行方法を問わず、アクセス許可を定義します。

アイデンティティベースのポリシー

アイデンティティベースのポリシーは、アイデンティティ (ユーザー、グループ、またはロール) にアタッチできる JSON アクセス許可ポリシードキュメントです。これらのポリシーは、アイデンティティがどのリソースに対してどのような条件下でどのようなアクションを実行できるかを制御します。アイデンティティベースポリシーの作成方法については、IAM ユーザーガイド の カスタマー管理ポリシーでカスタム IAM アクセス許可を定義する を参照してください。

アイデンティティベースのポリシーは、インラインポリシー (単一の ID に直接埋め込む) または管理ポリシー (複数の ID にアタッチされたスタンドアロンポリシー) にすることができます。管理ポリシーとインラインポリシーのいずれかを選択する方法については、「IAM ユーザーガイド」の「管理ポリシーとインラインポリシーのいずれかを選択する」を参照してください。

リソースベースのポリシー

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。例としては、IAM ロール信頼ポリシーや Amazon S3 バケットポリシーなどがあります。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。リソースベースのポリシーでは、プリンシパルを指定する必要があります。

リソースベースのポリシーは、そのサービス内にあるインラインポリシーです。リソースベースのポリシーでは、IAM の AWS マネージドポリシーを使用できません。

アクセスコントロールリスト (ACL)

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするためのアクセス許可を持つかを制御します。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

Amazon S3、および Amazon VPC は AWS WAF、ACLs。ACL の詳細については、Amazon Simple Storage Service デベロッパーガイド の アクセスコントロールリスト (ACL) の概要 を参照してください。

その他のポリシータイプ

AWS は、より一般的なポリシータイプによって付与されるアクセス許可の上限を設定できる追加のポリシータイプをサポートしています。

- アクセス許可の境界 – アイデンティティベースのポリシーで IAM エンティティに付与することのできるアクセス許可の数の上限を設定します。詳細については、「IAM ユーザーガイド」の「IAM エンティティのアクセス許可境界」を参照してください。
- サービスコントロールポリシー (SCP) - AWS Organizations内の組織または組織単位の最大のアクセス許可を指定します。詳細については、「AWS Organizations ユーザーガイド」の「サービスコントロールポリシー」を参照してください。
- リソースコントロールポリシー (RCP) – は、アカウント内のリソースで利用できる最大数のアクセス許可を定義します。詳細については、「AWS Organizations ユーザーガイド」の「リソースコントロールポリシー (RCP)」を参照してください。
- セッションポリシー – ロールまたはフェデレーションユーザーの一時セッションを作成する際にパラメータとして渡される高度なポリシーです。詳細については、「IAM ユーザーガイド」の「セッションポリシー」を参照してください。

複数のポリシータイプ

1 つのリクエストに複数のタイプのポリシーが適用されると、結果として作成されるアクセス許可を理解するのがさらに難しくなります。が複数のポリシータイプが関与する場合にリクエストを許可するかどうか AWS を決定する方法については、「IAM ユーザーガイド」の「ポリシー評価ロジック」を参照してください。

Amazon MQ で IAM が機能する仕組み

IAM を使用して Amazon MQ へのアクセスを管理する前に、Amazon MQ で使用できる IAM 機能について理解しておく必要があります。Amazon MQ およびその他の AWS のサービスが IAM と連携する方法の概要については、IAM ユーザーガイドのAWS 「IAM と連携する のサービス」を参照してください。

Amazon MQ はAmazon MQ API オペレーションに IAM を使用してブローカーを作成、更新、削除、一覧表示します。メッセージを発行およびサブスクライブするためのブローカーアクセスのために、Amazon MQ for ActiveMQ はネイティブ ActiveMQ 認証と LDAP をサポートし、Amazon MQ for RabbitMQ は IAM 認証やその他の方法をサポートします。詳細については、「the section called “ブローカーの認証と認可”」を参照してください。

トピック

- Amazon MQ のアイデンティティベースポリシー
- Amazon MQ のリソースベースポリシー
- Amazon MQ タグに基づいた認可
- Amazon MQ の IAM ロール

Amazon MQ のアイデンティティベースポリシー

IAM アイデンティティベースポリシーでは、許可または拒否するアクションとリソース、またアクションを許可または拒否する条件を指定できます。Amazon MQ は、特定のアクション、リソース、および条件キーをサポートしています。JSON ポリシーで使用するすべての要素については、「IAM ユーザーガイド」の「IAM JSON ポリシー要素のリファレンス」を参照してください。

アクション

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

JSON ポリシーの `Action` 要素にはポリシー内のアクセスを許可または拒否するために使用できるアクションが記述されます。このアクションは関連付けられたオペレーションを実行するためのアクセス許可を付与するポリシーで使用されます。

Amazon MQ のポリシーアクションは、アクションの前にプレフィックス `mq:` を使用します。例えば、Amazon MQ `CreateBroker` API オペレーションで Amazon MQ インスタンスを実行する許可を付与するには、ユーザーのポリシーに `mq:CreateBroker` アクションを含めます。ポリシーステートメントには、`Action` または `NotAction` エレメントを含める必要があります。Amazon MQ は、このサービスで実行できるタスクを記述する独自のアクションのセットを定義します。

単一のステートメントに複数のアクションを指定するには次のようにコンマで区切ります。

```
"Action": [
      "mq:action1",
      "mq:action2"
```

ワイルドカード (*) を使用して複数アクションを指定できます。例えば、`Describe` という単語で始まるすべてのアクションを指定するには、次のアクションを含めます。

```
"Action": "mq:Describe*"
```

Amazon MQ アクションのリストを確認するには、IAM ユーザーガイドの「Amazon MQ で定義されるアクション」を参照してください。

リソース

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

Resource JSON ポリシー要素はアクションが適用されるオブジェクトを指定します。ベストプラクティスとして、Amazon リソースネーム (ARN) を使用してリソースを指定します。リソースレベルのアクセス許可をサポートしないアクションの場合は、ステートメントがすべてのリソースに適用されることを示すために、ワイルドカード (*) を使用します。

```
"Resource": "*"
```

Amazon MQ では、プライマリ AWS リソースは Amazon MQ メッセージブローカーとその設定です。Amazon MQ ブローカーと設定には、以下の表にあるとおり、それぞれ一意の Amazon リソースネーム (ARN) が関連付けられています。

リソースタイプ	ARN	条件キー
brokers	arn:aws:mq:us-east-1:123456789012:broker:${brokerName}:${brokerId}	aws:ResourceTag/${TagKey}
configurations	arn:${Partition}:mq:${Region}:${Account}:configuration:${configuration-id}	aws:ResourceTag/${TagKey}

ARN の形式の詳細については、「Amazon リソースネーム (ARNs)」と AWS 「サービス名前空間」を参照してください。

例えば、ステートメントでブローカー ID b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9 を持つ MyBroker というブローカーを指定するには、次の ARN を使用します。

```
"Resource": "arn:aws:mq:us-
east-1:123456789012:broker:MyBroker:b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9"
```

特定のアカウントに属するすべてのブローカーと設定を指定するには、ワイルドカード (*) を使用します。

```
"Resource": "arn:aws:mq:us-east-1:123456789012:*"
```

リソースを作成するためのアクションなど、Amazon MQ アクションには特定のリソースで実行できないものがあります。このような場合はワイルドカード *を使用する必要があります。

```
"Resource": "*"
```

API アクション `CreateTags` には、ブローカーと設定の両方が必要です。複数リソースを単一ステートメントで指定するには、ARN をカンマで区切ります。

```
"Resource": [
      "resource1",
      "resource2"
```

Amazon MQ のリソースタイプとそれらの ARN のリストを確認するには、IAM ユーザーガイドの「Amazon MQ で定義されるリソースタイプ」を参照してください。どのアクションで各リソースの ARN を指定できるかについては、「Amazon MQ で定義されるアクション」を参照してください。

条件キー

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

`Condition` 要素は、定義された基準に基づいてステートメントが実行される時期を指定します。イコールや未満などの条件演算子を使用して条件式を作成して、ポリシーの条件とリクエスト内の値を一致させることができます。すべての AWS グローバル条件キーを確認するには、「IAM ユーザーガイド」のAWS 「グローバル条件コンテキストキー」を参照してください。

Amazon MQ はサービス固有の条件キーを定義しませんが、いくつかのグローバル条件キーの使用がサポートされています。Amazon MQ の条件キーのリストを確認するには、IAM ユーザーガイドの「Amazon MQ の条件キー」を参照してください。条件キーを使用できるアクションとリソースについては、「Amazon MQ で定義されるアクション」を参照してください。

条件キー	説明	タイプ
aws:RequestTag/${TagKey}	リクエストで渡されたタグに基づいてアクションをフィルタリングします。	String
aws:ResourceTag/${TagKey}	リソースに関連付けられているタグに基づいてアクションをフィルタリングします。	String
aws:TagKeys	リクエストで渡されたタグキーに基づいてアクションをフィルタリングします。	String

例

Amazon MQ のアイデンティティベースポリシーの例を確認するには、「Amazon MQ のアイデンティティベースポリシーの例」を参照してください。

Amazon MQ のリソースベースポリシー

現在、Amazon MQ はリソースベースの許可またはリソースベースのポリシーを使用した IAM 認証をサポートしていません。

Amazon MQ タグに基づいた認可

タグは、Amazon MQ リソースにアタッチする、または Amazon MQ へのリクエストで渡すことができます。タグに基づいてアクセスを管理するには、`mq:ResourceTag/key-name`、`aws:RequestTag/key-name`、または `aws:TagKeys` の条件キーを使用して、ポリシーの条件要素でタグ情報を提供します。

Amazon MQ はタグベースのポリシーをサポートしています。例えば、キー `environment` および値 `production` を持つタグが含まれる Amazon MQ リソースへのアクセスを拒否することができます。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
```

```
        {
            "Effect": "Deny",
            "Action": [
                "mq:DeleteBroker",
                "mq:RebootBroker",
                "mq:DeleteTags"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/environment": "production"
                }
            }
        }
    ]
}
```

このポリシーは、environment/production タグが含まれる Amazon MQ ブローカーを削除または再起動する能力を Deny します。

タグ付けの詳細については、以下を参照してください。

- Amazon MQ リソースへのタグの追加
- IAM タグを使用したアクセスの制御

Amazon MQ の IAM ロール

IAM ロールは、特定のアクセス許可を持つ AWS アカウント内のエンティティです。

Amazon MQ での一時的な認証情報の使用

一時的な認証情報を使用して、フェデレーションでサインインする、IAM 役割を引き受ける、またはクロスアカウント役割を引き受けることができます。一時的なセキュリティ認証情報を取得するには、AssumeRole や GetFederationToken などの AWS STS API オペレーションを呼び出します。

Amazon MQ は、一時的な認証情報の使用をサポートします。

サービス役割

この機能により、ユーザーに代わってサービスがサービスロールを引き受けることが許可されます。この役割により、サービスがお客様に代わって他のサービスのリソースにアクセスし、アクションを

完了することが許可されます。サービスロールはIAM アカウントに表示され、アカウントによって所有されます。つまり、IAM 管理者はこの役割の権限を変更できます。ただし、それにより、サービスの機能が損なわれる場合があります。

Amazon MQ は、サービスロールをサポートします。

Amazon MQ のアイデンティティベースポリシーの例

デフォルトでは、ユーザーとロールには Amazon MQ リソースを作成または変更するアクセス許可がありません。また、 AWS マネジメントコンソール、 AWS CLI、または AWS API を使用してタスクを実行することはできません。IAM 管理者は、ユーザーとロールに必要な、指定されたリソースで特定の API オペレーションを実行する権限をユーザーとロールに付与する IAM ポリシーを作成する必要があります。続いて、管理者はそれらの権限が必要な IAM ユーザーまたはグループにそのポリシーをアタッチする必要があります。

JSON ポリシードキュメントのこれらの例を使用して、IAM アイデンティティベースのポリシーを作成する方法については、「IAM ユーザーガイド」の「JSON タブでのポリシーの作成」を参照してください。

トピック

- ポリシーに関するベストプラクティス
- Amazon MQ コンソールの使用
- ユーザーが自分の許可を表示できるようにする

ポリシーに関するベストプラクティス

ID ベースのポリシーは、ユーザーのアカウント内で誰かが Amazon MQ リソースを作成、アクセス、または削除できるどうかを決定します。これらのアクションでは、 AWS アカウントに費用が発生する場合があります。アイデンティティベースポリシーを作成したり編集したりする際には、以下のガイドラインと推奨事項に従ってください:

- AWS 管理ポリシーを開始し、最小特権のアクセス許可に移行する – ユーザーとワークロードにアクセス許可の付与を開始するには、多くの一般的なユースケースにアクセス許可を付与するAWS 管理ポリシーを使用します。これらは で使用できます AWS アカウント。ユースケースに固有の AWS カスタマー管理ポリシーを定義することで、アクセス許可をさらに減らすことをお勧めします。詳細については、IAM ユーザーガイド の AWS マネージドポリシー または ジョブ機能のAWS マネージドポリシー を参照してください。

- 最小特権を適用する – IAM ポリシーでアクセス許可を設定する場合は、タスクの実行に必要な許可のみを付与します。これを行うには、特定の条件下で特定のリソースに対して実行できるアクションを定義します。これは、最小特権アクセス許可とも呼ばれています。IAM を使用して許可を適用する方法の詳細については、IAM ユーザーガイド の IAM でのポリシーとアクセス許可 を参照してください。
- IAM ポリシーで条件を使用してアクセスをさらに制限する - ポリシーに条件を追加して、アクションやリソースへのアクセスを制限できます。たとえば、ポリシー条件を記述して、すべてのリクエストを SSL を使用して送信するように指定できます。条件を使用して、サービスアクションが などの特定の を通じて使用されている場合に AWS のサービス、サービスアクションへのアクセスを許可することもできます CloudFormation。詳細については、IAM ユーザーガイド の IAM JSON ポリシー要素:条件 を参照してください。
- IAM アクセスアナライザー を使用して IAM ポリシーを検証し、安全で機能的な権限を確保する - IAM アクセスアナライザー は、新規および既存のポリシーを検証して、ポリシーが IAM ポリシー言語 (JSON) および IAM のベストプラクティスに準拠するようにします。IAM アクセスアナライザーは 100 を超えるポリシーチェックと実用的な推奨事項を提供し、安全で機能的なポリシーの作成をサポートします。詳細については、IAM ユーザーガイド の IAM Access Analyzer でポリシーを検証する を参照してください。
- 多要素認証 (MFA) を要求する – で IAM ユーザーまたはルートユーザーを必要とするシナリオがある場合は AWS アカウント、MFA をオンにしてセキュリティを強化します。API オペレーションが呼び出されるときに MFA を必須にするには、ポリシーに MFA 条件を追加します。詳細については、IAM ユーザーガイド の MFA を使用した安全な API アクセス を参照してください。

IAM でのベストプラクティスの詳細については、IAM ユーザーガイド の IAM でのセキュリティのベストプラクティス を参照してください。

Amazon MQ コンソールの使用

Amazon MQ コンソールにアクセスするには、許可の最小限のセットが必要です。これらのアクセス許可により、 AWS アカウントの Amazon MQ リソースの詳細を一覧表示および表示できます。最小限必要な許可よりも厳しく制限されたアイデンティティベースポリシーを作成すると、そのポリシーを添付したエンティティ (IAM ユーザーまたはロール) に対してコンソールが意図したとおりに機能しません。

これらのエンティティが引き続き Amazon MQ コンソールを使用できるようにするには、エンティティに次の AWS 管理ポリシーもアタッチします。詳細については、「IAM ユーザーガイド」の「ユーザーへのアクセス許可の追加」を参照してください。

```
AmazonMQReadOnlyAccess
```

AWS CLI または AWS API のみを呼び出すユーザーには、最小限のコンソールアクセス許可を付与する必要はありません。代わりに、実行しようとしている API オペレーションに一致するアクションのみへのアクセスが許可されます。

ユーザーが自分の許可を表示できるようにする

この例では、ユーザーアイデンティティにアタッチされたインラインおよびマネージドポリシーの表示を IAM ユーザーに許可するポリシーの作成方法を示します。このポリシーには、コンソールで、または AWS CLI または AWS API を使用してプログラムでこのアクションを実行するアクセス許可が含まれています。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
```

```
        }
    ]
}
```

Amazon MQ の API 認証と認可

Amazon MQ は API 認証に標準 AWS リクエスト署名を使用します。詳細については、『AWS』の「AWS 全般のリファレンス API リクエストの署名」を参照してください。

> **Note**
>
> 現在、Amazon MQ はリソースベースの許可またはリソースベースのポリシーを使用した IAM 認証をサポートしていません。

ブローカー、設定、 AWS およびユーザーの使用をユーザーに許可するには、IAM ポリシーのアクセス許可を編集する必要があります。

トピック

- Amazon MQ ブローカーを作成するために必要な IAM 許可
- Amazon MQ REST API 許可リファレンス
- Amazon MQ の追加のアクセス許可リファレンス
- Amazon MQ API アクションに対するリソースレベルの許可

Amazon MQ ブローカーを作成するために必要な IAM 許可

ブローカーを作成するには、`AmazonMQFullAccess` IAM ポリシーを使用するか、以下の EC2 許可を IAM ポリシーに含める必要があります。

以下のカスタムポリシーは、ActiveMQ ブローカーを作成するために Amazon MQ が必要とするリソースを操作するための許可を付与する 2 つのステートメント (1 つは条件付き) で構成されています。

> **Important**
>
> - `ec2:CreateNetworkInterface` アクションは、ユーザーに代わってアカウントに Elastic Network Interface (ENI) を作成することを Amazon MQ に許可するために必要です。

- `ec2:CreateNetworkInterfacePermission` アクションは、Amazon MQ が ENI を ActiveMQ ブローカーにアタッチすることを認可します。
- `ec2:AuthorizedService` 条件キーは、ENI 許可が Amazon MQ サービスアカウントのみに付与されることを確実にします。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Action": [
            "mq:*",
            "ec2:CreateNetworkInterface",
            "ec2:DeleteNetworkInterface",
            "ec2:DetachNetworkInterface",
            "ec2:DescribeInternetGateways",
            "ec2:DescribeNetworkInterfaces",
            "ec2:DescribeRouteTables",
            "ec2:DescribeSecurityGroups",
            "ec2:DescribeSubnets",
            "ec2:DescribeVpcs"
        ],
        "Effect": "Allow",
        "Resource": "*"
    },{
        "Action": [
            "ec2:CreateNetworkInterfacePermission",
            "ec2:DeleteNetworkInterfacePermission",
            "ec2:DescribeNetworkInterfacePermissions"
        ],
        "Effect": "Allow",
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "ec2:AuthorizedService": "mq.amazonaws.com"
            }
        }
    }]
}
```

詳細については、「ステップ 2: ユーザーを作成して AWS 認証情報を取得する」および「Amazon MQ Elastic Network Interface を変更または削除しない」を参照してください。

Amazon MQ REST API 許可リファレンス

以下の表には、Amazon MQ REST API と、それらに対応する IAM 許可がリストされています。

Amazon MQ REST API と必要な許可

Amazon MQ REST API	必要な許可
CreateBroker	mq:CreateBroker
CreateConfiguration	mq:CreateConfiguration
CreateTags	mq:CreateTags
CreateUser	mq:CreateUser
DeleteBroker	mq:DeleteBroker
DeleteUser	mq:DeleteUser
DescribeBroker	mq:DescribeBroker
DescribeConfiguration	mq:DescribeConfiguration
DescribeConfigurationRevision	mq:DescribeConfigurationRevision
DescribeUser	mq:DescribeUser
ListBrokers	mq:ListBrokers
ListConfigurationRevisions	mq:ListConfigurationRevisions
ListConfigurations	mq:ListConfigurations
ListTags	mq:ListTags
ListUsers	mq:ListUsers
RebootBroker	mq:RebootBroker

Amazon MQ REST API	必要な許可
UpdateBroker	mq:UpdateBroker
UpdateConfiguration	mq:UpdateConfiguration
UpdateUser	mq:UpdateUser

Amazon MQ の追加のアクセス許可リファレンス

次の表に、Amazon MQ API と、OAuth 2.0 認証などの特定の機能に必要な追加の IAM アクセス許可を示します。

Amazon MQ REST API	アクセス許可	説明
UpdateBroker	mq:UpdateBrokerAccessConfiguration	関連するブローカー設定で認証および認可オプションを更新するには、このアクセス許可が必要です。詳細については、「Amazon MQ for RabbitMQ に対する OAuth 2.0 の認証と認可」を参照してください。

Amazon MQ API アクションに対するリソースレベルの許可

リソースレベルの許可とは、ユーザーがアクションを実行できるリソースを指定する能力を意味します。Amazon MQ は、リソースレベルの許可を部分的にサポートします。特定の Amazon MQ アクションでは、満たす必要がある条件、またはユーザーが使用できる特定のリソースに基づいて、ユーザーにこれらのアクションの使用が許可されるタイミングを制御できます。

以下の表では、現在リソースレベルの許可をサポートしている Amazon MQ API アクションと、各アクションに対してサポートされるリソース、リソース ARN、条件キーを説明します。

> **Important**
>
> Amazon MQ API アクションがこの表に示されていない場合、そのアクションはリソースレベルの許可をサポートしていません。Amazon MQ API アクションがリソースレベルの許可をサポートしない場合、アクションを使用する許可をユーザーに付与できますが、ポリシーステートメントのリソース要素にワイルドカード (*) を指定する必要があります。

API アクション	リソースタイプ (* 必須)
CreateConfiguration	設定*
CreateTags	ブローカー、設定
CreateUser	ブローカー
DeleteBroker	ブローカー
DeleteUser	ブローカー
DescribeBroker	ブローカー
DescribeConfiguration	設定*
DescribeConfigurationRevision	設定*
DescribeUser	ブローカー
ListConfigurationRevisions	設定*
ListConfigurationRevisions	設定*
ListTags	ブローカー、設定
ListUsers	ブローカー
RebootBroker	ブローカー

API アクション	リソースタイプ (* 必須)
UpdateBroker	ブローカー
UpdateConfiguration	設定*
UpdateUser	ブローカー

ブローカーの認証と認可

Amazon MQ は、ブローカーエンジンのタイプに応じて異なる認証方法と認可方法を提供します。

Amazon MQ for ActiveMQ の認証と認可

Amazon MQ for ActiveMQ は、次の認証および認可方法をサポートしています。

シンプルな認証と認可

この方法では、ブローカーユーザーは Amazon MQ コンソールまたは API を使用して作成および管理されます。ユーザーには、キュー、トピック、および ActiveMQ ウェブコンソールにアクセスするための特定のアクセス許可を設定できます。この方法の詳細については、ActiveMQ ブローカーユーザーの作成」を参照してください。

LDAP 認証と認可

この方法では、ブローカーユーザーは LDAP サーバーに保存されている認証情報を使用して認証します。ユーザーを追加、削除、変更し、LDAP サーバーを介してトピックとキューにアクセス許可を割り当てることで、一元的な認証と認可を提供できます。この方法の詳細については、ActiveMQ ブローカーと LDAP の統合」を参照してください。

Amazon MQ for RabbitMQ の認証と認可

Amazon MQ for RabbitMQ は、次の認証および認可方法をサポートしています。

シンプルな認証と認可

この方法では、ブローカーユーザーは内部的に RabbitMQ ブローカーに保存され、ウェブコンソールまたは管理 API を介して管理されます。vhost、交換、キュー、トピックのアクセス許可は、RabbitMQ で直接設定されます。これがデフォルトの方法です。詳細については、「シンプルな認証と認可」を参照してください。

OAuth 2.0 の認証と認可

この方法では、ブローカーユーザーとそのアクセス許可は、外部 OAuth 2.0 ID プロバイダー (IdP) によって管理されます。vhost、交換、キュー、トピックのユーザー認証とリソースアクセス許可は、OAuth 2.0 プロバイダーのスコープシステムを通じて一元化されます。これにより、ユーザー管理が簡素化され、既存の ID システムとの統合が可能になります。詳細については、OAuth 2.0 の認証と認可」を参照してください。

IAM 認証と認可

この方法では、ブローカーユーザーは IAM AWS アウトバウンドフェデレーションを介して IAM 認証情報を使用して認証します。IAM 認証情報は、 AWS Security Token Service (STS) から JWT トークンを取得するために使用されます。これらの JWT トークンは、認証用の OAuth 2.0 トークンとして機能します。このメソッドは、Amazon MQ for RabbitMQ の既存の OAuth 2.0 サポートを活用します。 は OAuth 2.0 ID プロバイダー AWS として機能します。ユーザー認証は IAM AWS によって処理され、vhost、エクスチェンジ、キュー、トピックのリソースアクセス許可は RabbitMQ で設定された IAM ポリシーとスコープエイリアスによって管理されます。詳細については、「IAM 認証と認可」を参照してください。

LDAP 認証と認可

この方法では、ブローカーユーザーとそのアクセス許可は外部 LDAP ディレクトリサービスによって管理されます。ユーザー認証とリソースのアクセス許可は LDAP サーバーを通じて一元化されるため、ユーザーは既存のディレクトリサービス認証情報を使用して RabbitMQ にアクセスできます。詳細については、「LDAP 認証と認可」を参照してください。

HTTP 認証と認可

この方法では、ブローカーユーザーとそのアクセス許可は外部 HTTP サーバーによって管理されます。ユーザー認証とリソースのアクセス許可は HTTP サーバーを通じて一元化されるため、ユーザーは独自の認証および認可プロバイダーを使用して RabbitMQ にアクセスできます。この方法の詳細については、「HTTP 認証と認可」を参照してください。

SSL 証明書認証

Amazon MQ は、RabbitMQ ブローカーの相互 TLS (mTLS) をサポートしています。SSL 認証プラグインは、mTLS 接続からのクライアント証明書を使用してユーザーを認証します。この方法では、ブローカーユーザーはユーザー名とパスワードの認証情報の代わりに X.509 クライアント証明書を使用して認証されます。クライアントの証明書は信頼できる認証機関 (CA) に対して検証され、ユーザー名は共通名 (CN) やサブジェクト代替名 (SAN) などの証明書のフィールドから抽出されま

す。この方法は、ネットワーク経由で認証情報を送信せずに強力な認証を提供します。詳細については、「SSL 証明書認証」を参照してください。

> **Note**
>
> RabbitMQ は、同時に使用する複数の認証および認可方法をサポートしています。たとえば、OAuth 2.0 と簡易 (内部) 認証の両方を有効にできます。詳細については、OAuth 2.0 チュートリアルセクション「OAuth 2.0 と簡易 (内部) 認証の両方を有効にする」および「RabbitMQ アクセスコントロールドキュメント」を参照してください。
> Amazon MQ では、認証設定をテストするときに内部ユーザーを作成することをお勧めします。これにより、RabbitMQ 管理 API を使用してアクセス設定を検証できます。詳細については、「アクセス検証」を参照してください。

AWS Amazon MQ の マネージドポリシー

AWS 管理ポリシーは、 によって作成および管理されるスタンドアロンポリシーです AWS。 AWS 管理ポリシーは、ユーザー、グループ、ロールにアクセス許可の割り当てを開始できるように、多くの一般的なユースケースにアクセス許可を付与するように設計されています。

AWS 管理ポリシーは、すべての AWS お客様が使用できるため、特定のユースケースに対して最小特権のアクセス許可を付与しない場合があることに注意してください。ユースケースに固有のカスタマー管理ポリシーを定義して、アクセス許可を絞り込むことをお勧めします。

AWS 管理ポリシーで定義されているアクセス許可は変更できません。が AWS マネージドポリシーで定義されたアクセス許可 AWS を更新すると、ポリシーがアタッチされているすべてのプリンシパル ID (ユーザー、グループ、ロール) に影響します。 AWS は、新しい が起動されるか、新しい API オペレーション AWS のサービス が既存のサービスで使用できるようになったときに、 AWS マネージドポリシーを更新する可能性が最も高くなります。

詳細については、「IAM ユーザーガイド」の「AWS マネージドポリシー」を参照してください。

Amazon MQ は、以下の AWS 管理ポリシーをサポートしています。

- AmazonMQApiFullAccess
- AmazonMQApiReadOnlyAccess
- AmazonMQFullAccess
- AmazonMQReadOnlyAccess
- AmazonMQServiceRolePolicy

AWS マネージドポリシー: AmazonMQServiceRolePolicy

IAM エンティティに AmazonMQServiceRolePolicy をアタッチすることはできません。このポリシーは、Amazon MQ がユーザーに代わってアクションを実行することを許可するサービスリンクロールにアタッチされます。この許可ポリシーと、それが Amazon MQ に実行を許可するアクションの詳細については、「the section called “Amazon MQ のサービスリンクロール許可”」を参照してください。

AWS マネージドポリシーに対する Amazon MQ の更新

このサービスがこれらの変更の追跡を開始してからの Amazon MQ の AWS マネージドポリシーの更新に関する詳細を表示します。このページへの変更に関する自動アラートを受け取るには、Amazon MQ のドキュメント履歴ページで RSS フィードにサブスクライブしてください。

変更	説明	日付
Amazon MQ が変更の追跡を開始しました。	Amazon MQ は、 AWS 管理ポリシーの変更の追跡を開始しました。	2021 年 5 月 5 日

Amazon MQ のサービスリンクロールの使用

Amazon MQ は AWS Identity and Access Management (IAM) サービスにリンクされたロールを使用します。サービスリンクロールは、Amazon MQ に直接リンクされた特殊なタイプの IAM ロールです。サービスにリンクされたロールは Amazon MQ によって事前定義されており、サービスがユーザーに代わって他の AWS サービスを呼び出すために必要なすべてのアクセス許可が含まれています。

必要な許可を手動で追加する必要がないため、サービスリンクロールは Amazon MQ のセットアップを容易にします。サービスリンクロールの許可は Amazon MQ が定義し、別段の定義がない限り、Amazon MQ のみがそのロールを引き受けることができます。定義される許可は信頼ポリシーと許可ポリシーに含まれており、その許可ポリシーを他の IAM エンティティにアタッチすることはできません。

サービスリンクロールを削除するには、最初に関連リソースを削除する必要があります。これは、リソースにアクセスするための許可を誤って削除できないため、Amazon MQ リソースを保護します。

サービスにリンクされたロールをサポートするその他のサービスについては、「IAM と連携する AWS のサービス」を参照し、[Service-Linked Role] (サービスにリンクされたロール) 列が [Yes] (はい) になっているサービスを検索してください。サービスリンクロールに関するドキュメントをサービスで表示するには、[Yes] (はい) リンクを選択します。

Amazon MQ のサービスリンクロール許可

Amazon MQ は、AWSServiceRoleForAmazonMQ という名前のサービスにリンクされたロールを使用します。Amazon MQ は、このサービスにリンクされたロールを使用してユーザーに代わって AWS サービスを呼び出します。

AWSServiceRoleForAmazonMQ サービスリンクロールは、ロールの引き受けに以下のサービスを信頼します。

- `mq.amazonaws.com`

Amazon MQ は、指定されたリソースで以下のアクションを完了するために、AWSServiceRoleForAmazonMQ サービスリンクロールにアタッチされる許可ポリシー `AmazonMQServiceRolePolicy` を使用します。

- アクション: `vpc` リソースでの `ec2:CreateVpcEndpoint` アクション。
- アクション: `subnet` リソースでの `ec2:CreateVpcEndpoint` アクション。
- アクション: `security-group` リソースでの `ec2:CreateVpcEndpoint` アクション。
- アクション: `vpc-endpoint` リソースでの `ec2:CreateVpcEndpoint` アクション。
- アクション: `vpc` リソースでの `ec2:DescribeVpcEndpoints` アクション。
- アクション: `subnet` リソースでの `ec2:DescribeVpcEndpoints` アクション。
- アクション: `vpc-endpoint` リソースでの `ec2:CreateTags` アクション。
- アクション: `log-group` リソースでの `logs:PutLogEvents` アクション。

- アクション: log-group リソースでの logs:DescribeLogStreams アクション。
- アクション: log-group リソースでの logs:DescribeLogGroups アクション。
- アクション: log-group リソースでの CreateLogStream アクション。
- アクション: log-group リソースでの CreateLogGroup アクション。

Amazon MQ for RabbitMQ ブローカーの作成時、AmazonMQServiceRolePolicy 許可ポリシーは、Amazon MQ がユーザーに代わって以下のタスクを実行することを許可します。

- ユーザー指定の Amazon VPC、サブネット、およびセキュリティグループを使用して、ブローカーの Amazon VPC エンドポイントを作成する。ブローカー用に作成されたエンドポイントは、RabbitMQ マネジメントコンソール、Management API、またはプログラム経由でブローカーに接続するために使用できます。
- ロググループを作成して、ブローカーログを Amazon CloudWatch Logs に発行する。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeVpcEndpoints"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateVpcEndpoint"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:vpc/*",
                "arn:aws:ec2:*:*:subnet/*",
                "arn:aws:ec2:*:*:security-group/*"
```

```
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateVpcEndpoint"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:vpc-endpoint/*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/AMQManaged": "true"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": "arn:aws:ec2:*:*:vpc-endpoint/*",
            "Condition": {
                "StringEquals": {
                    "ec2:CreateAction": "CreateVpcEndpoint"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DeleteVpcEndpoints"
            ],
            "Resource": "arn:aws:ec2:*:*:vpc-endpoint/*",
            "Condition": {
                "StringEquals": {
                    "ec2:ResourceTag/AMQManaged": "true"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents",
```

```
                "logs:DescribeLogStreams",
                "logs:DescribeLogGroups",
                "logs:CreateLogStream",
                "logs:CreateLogGroup"
            ],
            "Resource": [
                "arn:aws:logs:*:*:log-group:/aws/amazonmq/*"
            ]
        }
    ]
}
```

サービスリンクロールの作成、編集、削除を IAM エンティティ (ユーザー、グループ、ロールなど) に許可するには、許可を設定する必要があります。詳細については、IAM ユーザーガイドの「サービスリンクロールの許可」を参照してください。

Amazon MQ のサービスリンクロールの作成

サービスリンクロールを手動で作成する必要はありません。ブローカーを初めて作成すると、Amazon MQ はユーザーに代わって AWS サービスを呼び出すサービスにリンクされたロールを作成します。その後作成するすべてのブローカーには同じロールが使用され、新しいロールは作成されません。

> **Important**
>
> このサービスリンクロールがアカウントに表示されるのは、このロールでサポートされている機能を使用する別のサービスでアクションが完了した場合です。詳細については、「IAM アカウントに新しいロールが表示される」を参照してください。

このサービスリンクロールを削除した後で再度作成する必要が生じた場合は、同じ手順でアカウントにロールを再作成できます。

IAM コンソールを使用して、Amazon MQ ユースケースでサービスリンクロールを作成することもできます。 AWS CLI または AWS API で、サービス名を使用して`mq.amazonaws.com`サービスにリンクされたロールを作成します。詳細については、IAM ユーザーガイドの「サービスリンクロールの作成」を参照してください。このサービスリンクロールを削除しても、同じ方法でロールを再作成できます。

> **Important**
>
> サービスリンクロールは、Amazon MQ for RabbitMQ に対してのみ作成されます。

Amazon MQ のサービスリンクロールの編集

Amazon MQ は、AWSServiceRoleForAmazonMQ サービスリンクロールの編集を許可しません。ただし、IAM を使用してロールの説明を編集することはできます。詳細については、IAM ユーザーガイドの「サービスリンクロールの編集」を参照してください。

Amazon MQ のサービスリンクロールの削除

サービスリンクロールを必要とする機能やサービスが不要になった場合は、ロールを削除することをお勧めします。そうすることで、積極的にモニタリングまたは保守されていない未使用のエンティティを排除できます。ただし、手動で削除する前に、サービスリンクロールのリソースをクリーンアップする必要があります。

> **Note**
>
> リソースを削除しようとしているときに Amazon MQ サービスがロールを使用している場合は、削除が失敗する可能性があります。失敗した場合は数分待ってから操作を再試行してください。

AWSServiceRoleForAmazonMQ が使用する Amazon MQ リソースを削除する

- AWS マネジメントコンソール、Amazon MQ CLI、または Amazon MQ API を使用して Amazon MQ ブローカーを削除します。ブローカーの削除の詳細については、「???」を参照してください。

サービスリンクロールを IAM で手動削除するには

IAM コンソール、 AWS CLI、または AWS API を使用して、AWSServiceRoleForAmazonMQ サービスにリンクされたロールを削除します。詳細については、IAM ユーザーガイドの「サービスリンクロールの削除」を参照してください。

Amazon MQ サービスリンクロールがサポートされるリージョン

Amazon MQ は、このサービスを利用できるすべてのリージョンでサービスリンクロールの使用をサポートします。詳細については、「AWS リージョンとエンドポイント」を参照してください。

リージョン名	リージョン識別子	Amazon MQ でのサポート
米国東部 (バージニア北部)	us-east-1	はい
米国東部 (オハイオ)	us-east-2	はい
米国西部 (北カリフォルニア)	us-west-1	はい
米国西部 (オレゴン)	us-west-2	はい
アジアパシフィック (ムンバイ)	ap-south-1	はい
アジアパシフィック (大阪)	ap-northeast-3	はい
アジアパシフィック (ソウル)	ap-northeast-2	はい
アジアパシフィック (シンガポール)	ap-southeast-1	はい
アジアパシフィック (シドニー)	ap-southeast-2	はい
アジアパシフィック (東京)	ap-northeast-1	はい
カナダ (中部)	ca-central-1	はい
欧州 (フランクフルト)	eu-central-1	はい
欧州 (アイルランド)	eu-west-1	はい
欧州 (ロンドン)	eu-west-2	はい
欧州 (パリ)	eu-west-3	はい
南米 (サンパウロ)	sa-east-1	はい
AWS GovCloud (US)	us-gov-west-1	いいえ

Amazon MQ アイデンティティとアクセスのトラブルシューティング

以下の情報を使用して、Amazon MQ と IAM の使用時に発生する可能性がある一般的な問題の診断と修正に役立てます。

トピック

- Amazon MQ でアクションを実行する認可がない
- iam:PassRole を実行する権限がない
- AWS アカウント以外のユーザーに Amazon MQ リソースへのアクセスを許可したい

Amazon MQ でアクションを実行する認可がない

にアクションを実行する権限がないと AWS マネジメントコンソール 通知された場合は、管理者に連絡してサポートを依頼する必要があります。管理者は、サインイン認証情報を提供した担当者です。

以下の例のエラーは、mateojackson ユーザーがコンソールを使用して、######の詳細を表示しようとしましたが、mq:*GetWidget* アクセス許可がない場合に発生します。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 mq:GetWidget on resource: my-example-widget
```

この場合、Mateo は、mq:*GetWidget* アクションを使用して *my-example-widget* リソースにアクセスできるように、管理者にポリシーの更新を依頼します。

iam:PassRole を実行する権限がない

iam:PassRole アクションを実行する権限がないというエラーが表示された場合は、ポリシーを更新して Amazon MQ にロールを渡せるようにする必要があります。

一部の AWS のサービス では、新しいサービスロールまたはサービスにリンクされたロールを作成する代わりに、既存のロールをそのサービスに渡すことができます。そのためには、サービスにロールを渡すアクセス許可が必要です。

以下のエラー例は、marymajor という名前の IAM ユーザーが、コンソールを使用して Amazon MQ でアクションを実行しようするときに発生します。ただし、このアクションをサービスが実行するには、サービスロールから付与されたアクセス許可が必要です。Mary には、ロールをサービスに渡すアクセス許可がありません。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

この場合、Mary のポリシーを更新してメアリーに `iam:PassRole` アクションの実行を許可する必要があります。

サポートが必要な場合は、 AWS 管理者にお問い合わせください。サインイン資格情報を提供した担当者が管理者です。

AWS アカウント以外のユーザーに Amazon MQ リソースへのアクセスを許可したい

他のアカウントのユーザーや組織外の人が、リソースにアクセスするために使用できるロールを作成できます。ロールの引き受けを委託するユーザーを指定できます。リソースベースのポリシーまたはアクセスコントロールリスト (ACL) をサポートするサービスの場合、それらのポリシーを使用して、リソースへのアクセスを付与できます。

詳細については、以下を参照してください:

- Amazon MQ がこれらの機能をサポートしているかどうかを確認するには、「Amazon MQ で IAM が機能する仕組み」を参照してください。
- 所有 AWS アカウント している のリソースへのアクセスを提供する方法については、「IAM ユーザーガイド」の「所有 AWS アカウント している別の の IAM ユーザーへのアクセスを提供する」を参照してください。
- リソースへのアクセスをサードパーティーに提供する方法については AWS アカウント、IAM ユーザーガイドの「サードパーティー AWS アカウント が所有する へのアクセスを提供する」を参照してください。
- ID フェデレーションを介してアクセスを提供する方法については、IAM ユーザーガイド の 外部で認証されたユーザー (ID フェデレーション) へのアクセスの許可 を参照してください。
- クロスアカウントアクセスにおけるロールとリソースベースのポリシーの使用方法の違いについては、「IAM ユーザーガイド」の「IAM でのクロスアカウントのリソースへのアクセス」を参照してください。

Amazon MQ のコンプライアンス検証

サードパーティーの監査者は、複数のコンプライアンスプログラムの一環として Amazon MQ のセキュリティと AWS コンプライアンスを評価します。このプログラムには、SOC、PCI、HIPAA などを含みます。

AWS のサービス が特定のコンプライアンスプログラムの範囲内にあるかどうかを確認するには、「コンプライアンスAWS のサービス プログラムによるスコープ」の「コンプライアンス」を参照して、関心のあるコンプライアンスプログラムを選択します。一般的な情報については、AWS「コンプライアンスプログラム」を参照してください。

を使用して、サードパーティーの監査レポートをダウンロードできます AWS Artifact。詳細については、「Downloading Reports in AWS Artifact」を参照してください。

を使用する際のお客様のコンプライアンス責任 AWS のサービス は、お客様のデータの機密性、貴社のコンプライアンス目的、適用される法律および規制によって決まります。を使用する際のコンプライアンス責任の詳細については AWS のサービス、AWS「 セキュリティドキュメント」を参照してください。

Amazon MQ の耐障害性

AWS のグローバルインフラストラクチャは AWS リージョンとアベイラビリティーゾーンを中心に構築されます。AWSリージョンには、低レイテンシー、高いスループット、そして高度の冗長ネットワークで接続されている複数の物理的に独立および隔離されたアベイラビリティーゾーンがあります。アベイラビリティーゾーンでは、ゾーン間で中断することなく自動的にフェイルオーバーするアプリケーションとデータベースを設計および運用することができます。アベイラビリティーゾーンは、従来の単一または複数のデータセンターインフラストラクチャよりも可用性、耐障害性、および拡張性が優れています。

AWS リージョンとアベイラビリティーゾーンの詳細については、AWS グローバルインフラストラクチャを参照してください。

Amazon MQ のインフラストラクチャセキュリティ

マネージドサービスである Amazon MQ は、 AWS グローバルネットワークセキュリティで保護されています。 AWS セキュリティサービスと がインフラストラクチャ AWS を保護する方法については、AWS「 クラウドセキュリティ」を参照してください。インフラストラクチャセキュリティのベストプラクティスを使用して環境を AWS 設計するには、「Security Pillar AWS Well-Architected Framework」の「Infrastructure Protection」を参照してください。

AWS 公開された API コールを使用して、ネットワーク経由で Amazon MQ にアクセスします。クライアントは次をサポートする必要があります。

- Transport Layer Security (TLS)。TLS 1.2 が必須で、TLS 1.3 をお勧めします。

- DHE (楕円ディフィー・ヘルマン鍵共有) や ECDHE (楕円曲線ディフィー・ヘルマン鍵共有) などの完全前方秘匿性 (PFS) による暗号スイート。これらのモードは、Java 7 以降など、最近のほとんどのシステムでサポートされています。

Amazon MQ のセキュリティベストプラクティス

以下の設計パターンは、Amazon MQ ブローカーのセキュリティを向上させることができます。

トピック

Amazon MQ がデータを暗号化する方法、およびサポートされるプロトコルのリストの詳細については、「データ保護」を参照してください。

パブリックアクセスビリティのないブローカーを優先する

パブリックアクセシビリティなしで作成されたブローカーには、VPC 外からアクセスできません。これにより、ブローカーがパブリックインターネットからの分散サービス妨害 (DDoS) 攻撃を受ける可能性が大幅に低減されます。詳細については、 AWS セキュリティブログの「攻撃領域を減らして DDoS 攻撃に備える方法」を参照してください。

認可マップを常に設定する

デフォルトでは、ActiveMQ には承認された認可マップがないため、認証されたすべてのユーザーが、ブローカーであらゆるアクションを実行することができます。したがって、グループごとにアクセス許可を制限することがベストプラクティスとなります。詳細については、「`authorizationEntry`」を参照してください。

> **Important**
>
> `activemq-webconsole` グループが含まれない認可マップを指定する場合、Amazon MQ ブローカーにメッセージを送信する権限、またはブローカーからメッセージを受信する権限がグループにないことから、ActiveMQ ウェブコンソールは使用できません。

VPC セキュリティグループを使用して不要なプロトコルをブロックする

プライベートブローカーのセキュリティを向上させるには、Amazon VPC セキュリティグループを正しく設定して、不要なプロトコルとポートの接続を制限する必要があります。例えば、OpenWire および ウェブコンソールへのアクセスを許可する一方で、ほとんどのプロトコルへのアクセスを制限するには、61617 および 8162 へのアクセスのみを許可することができます。これは、OpenWire とウェブコンソールが正常に機能することを可能にしながら、使用していないプロトコルをブロックすることによって、露出を制限します。

使用しているプロトコルポートのみを許可します。

- AMQP: 5671
- MQTT: 8883
- OpenWire: 61617
- STOMP: 61614
- WebSocket: 61619

詳細については、以下を参照してください。

- VPC のセキュリティグループ
- VPC のデフォルトセキュリティグループ
- セキュリティグループを操作する

Amazon MQ ブローカーのロギングとモニタリング

モニタリングは、 AWS ソリューションの信頼性、可用性、パフォーマンスを維持する上で重要な部分です。マルチポイント障害が発生した場合は、その障害をより簡単にデバッグできるように、 AWS ソリューションのすべての部分からモニタリングデータを収集する必要があります。 には、Amazon MQ リソースをモニタリングし、潜在的なインシデントに対応するための複数のツール AWS が用意されています。

CloudWatch を使用して、Amazon MQ ブローカーのメトリクスを表示および分析できます。ブローカーメトリクスは、CloudWatch コンソール、、 AWS CLIまたは CloudWatch AWS CLIから表示および分析できます。Amazon MQ 向けの CloudWatch メトリクスは、1 分おきにブローカーから自動的にポーリングされ、その後 CloudWatch にプッシュされます。ActiveMQ ブローカーの場合、CloudWatch は最初の 1000 個の送信先のみをモニタリングします。RabbitMQ ブローカーの場合、CloudWatch はコンシューマーの数順に並べられた最初 500 個の送信先のみをモニタリングします。

Amazon MQ メトリクスの完全なリストについては、「Amazon MQ for ActiveMQ ブローカーで利用可能な CloudWatch メトリクス」を参照してください。

メトリクスに対する CloudWatch アラームの作成については、Amazon CloudWatch ユーザーガイドで Amazon CloudWatch アラームの作成と編集を参照してください。

Amazon MQ 向けの CloudWatch メトリクスへのアクセス

CloudWatch メトリクスには AWS マネジメントコンソール、、 AWS CLI、および API を使用してアクセスできます。

を使用せずに CloudWatch メトリクスにアクセスすることもできます AWS マネジメントコンソール。

を使用して Amazon MQ メトリクスにアクセスするには AWS CLI、`get-metric-statistics` コマンドを使用します。詳細については、Amazon CloudWatch ユーザーガイドの「メトリクスの統計の取得」を参照してください。

CloudWatch API を使用して Amazon MQ メトリクスにアクセスするには、`GetMetricStatistics` アクションを使用します。詳細については、Amazon CloudWatch ユーザーガイドの「メトリクスの統計の取得」を参照してください。

を使用した CloudWatch メトリクスの取得 AWS マネジメントコンソール

次の例は、 AWS マネジメントコンソールを使用して Amazon MQ の CloudWatch メトリクスにアクセスする方法を示しています。Amazon MQ コンソールに既にサインインしている場合は、ブローカーの [詳細] ページで、[アクション]、[CloudWatch メトリクスの表示] の順に選択します。

1. CloudWatchコンソールにサインインします。
2. ナビゲーションパネルで [Metrics] を選択します。
3. [AmazonMQ] メトリクスの名前空間を選択します。
4. 次のいずれかのメトリクスディメンションを選択します。

   - ブローカーのメトリクス
   - ブローカー別のキューメトリクス
   - ブローカー別のトピックメトリクス

   この例では、[ブローカーのメトリクス] が選択されています。
5. これで、Amazon MQ メトリクスを調べることができるようになりました。

   - メトリクスを並べ替えるには、列見出しを使用します。

   - メトリクスをグラフ表示するには、メトリクスの横にあるチェックボックスを選択します。
   - メトリクスでフィルタするには、メトリクスの名前を選択し、[Add to search] を選択します。

Amazon MQ for ActiveMQ ブローカーで利用可能な CloudWatch メトリクス

Amazon MQ for ActiveMQ メトリクス

メトリクス	単位	説明
`AmqpMaximumConnections`	カウント	AMQP を使用してブローカーに接続できるクライアントの最大数。接続クォータの詳細については、「Quotas in

メトリクス	単位	説明
		Amazon MQ」を参照してください。
BurstBalance	割合 (%)	スループット最適化ブローカーのメッセージデータを永続化するために使用される Amazon EBS ボリュームに残っているバーストクレジットの割合 (%)。この残量がゼロになると、バーストバランスが補充されるまで、Amazon EBS ボリューム提供の IOPS が減少します。Amazon EBS でのバーストバランスの仕組みに関する詳細については、「I/O クレジットおよびバーストパフォーマンス」を参照してください。

メトリクス	単位	説明
CpuCreditBalance	クレジット (vCPU 分)	**Important** このメトリクスは、mq.t2.micro ブローカーインスタンスタイプでのみ使用できます。 CPU クレジットメトリクスは、5 分間隔でのみ利用可能です。 インスタンスが起動または開始後に蓄積した獲得 CPU クレジットの数 (起動クレジットの数を含む)。クレジット残高は、ブローカーインスタンスがそのベースライン CPU 使用率を超えてバーストするために消費できます。 クレジットは、獲得後にクレジット残高に蓄積され、消費後にクレジット残高から削除されます。クレジット残高には上限があります。制限に到達すると、新しく獲得されたクレジットはすべて破棄されます。
CpuUtilization	割合 (%)	割り当てられた Amazon EC2 コンピュートユニット (ECU) のうち、現在ブローカーが使用しているユニットの割合。

メトリクス	単位	説明
CurrentConnectionsCount	カウント	現在のブローカーでのアクティブな接続の現在の数。
EstablishedConnectionsCount	カウント	ブローカーで確立された、アクティブと非アクティブな接続の合計数。
HeapUsage	割合 (%)	ブローカーが現在使用している ActiveMQ JVM メモリ制限の割合。
InactiveDurableTopicSubscribersCount	カウント	非アクティブな永続トピックサブスクライバーの数 (最大 2000)。
JobSchedulerStorePercentUsage	割合 (%)	ジョブスケジューラストアで使用するディスク領域の割合 (%)。
JournalFilesForFastRecovery	カウント	クリーンシャットダウン後に再生されるジャーナルファイルの数。
JournalFilesForFullRecovery	カウント	クリーンでないシャットダウン後に再生されるジャーナルファイルの数。
MqttMaximumConnections	カウント	MQTT を使用してブローカーに接続できるクライアントの最大数。接続クォータの詳細については、「Quotas in Amazon MQ」を参照してください。

メトリクス	単位	説明
NetworkConnectorConnectionCount	カウント	NetworkConnector を使用してブローカーのネットワーク内のブローカーに接続されているノードの数。
NetworkIn	バイト	ブローカーの受信トラフィックのボリューム。
NetworkOut	バイト	ブローカーの送信トラフィックのボリューム。
OpenTransactionCount	カウント	進行中のトランザクションの総数。
OpenwireMaximumConnections	カウント	OpenWire を使用してブローカーに接続できるクライアントの最大数。接続クォータの詳細については、「Quotas in Amazon MQ」を参照してください。
StompMaximumConnections	カウント	STOMP を使用してブローカーに接続できるクライアントの最大数。接続クォータの詳細については、「Quotas in Amazon MQ」を参照してください。
StorePercentUsage	割合 (%)	ストレージ制限によって使用されている割合。これが 100 に達すると、ブローカーはメッセージを拒否します。
TempPercentUsage	割合 (%)	非永続的メッセージで使用可能な一時ストレージの割合 (%)。

メトリクス	単位	説明
TotalConsumerCount	カウント	現在のブローカーの送信先にサブスクライブされたメッセージコンシューマーの数。
TotalMessageCount	カウント	ブローカーに保存されたメッセージの数。
TotalProducerCount	カウント	現在のブローカーの送信先でのアクティブなメッセージプロデューサーの数。
VolumeReadOps	カウント	Amazon EBS ボリュームで実行された読み取り操作の数。
VolumeWriteOps	カウント	Amazon EBS ボリュームで実行された書き込み操作の数。
WsMaximumConnections	カウント	WebSocket を使用してブローカーに接続できるクライアントの最大数。接続クォータの詳細については、「Quotas in Amazon MQ」を参照してください。

ActiveMQ ブローカーメトリクスのディメンション

ディメンション	説明
Broker	ブローカーの名前 **Note** 単一インスタンスブローカーにはサフィックス -1 が付いています。高可用性対応のアクティブ/スタンバイブロー

ディメンション	説明
	カーには、その冗長ペアにサフィックス -1 と -2 が付いています。

ActiveMQ の送信先 (キューとトピック) メトリクス

> **Important**
>
> 以下のメトリクスには、CloudWatch のポーリング期間中の 1 分あたりの数が含まれます。
>
> - `EnqueueCount`
> - `ExpiredCount`
> - `DequeueCount`
> - `DispatchCount`
> - `InFlightCount`
>
> 例えば、5 分間の CloudWatch 期間では、`EnqueueCount` に 5 つの計数値があり、それぞれがその期間の 1 分間に対応します。`Maximum` および `Minimum` 統計は、指定した期間内の 1 分あたりの最小値と最大値を提供します。

メトリクス	単位	説明
`ConsumerCount`	カウント	送信先にサブスクライブされる消費者の数。
`EnqueueCount`	カウント	送信先に送信されるメッセージの数 (1 分あたり)。
`EnqueueTime`	時間 (ミリ秒)	メッセージがブローカーに届いてからコンシューマーに配信されるまでの、エンドツーエンドのレイテンシー。

メトリクス	単位	説明
		Note EnqueueTime は、プロデューサーがメッセージを送信してから、それがブローカーに到達するまでのエンドツーエンドレイテンシーを測定しません。また、ブローカーがメッセージを受信してから、ブローカーがそれを承認するまでのレイテンシーも測定しません。EnqueueTime は、ブローカーがメッセージを受信した瞬間から、コンシューマーに正常に配信されるまでのミリ秒数です。
ExpiredCount	カウント	期限切れのために配信できなかったメッセージの数 (1 分あたり)。
DispatchCount	カウント	コンシューマーに送信されたメッセージの数 (1 分あたり)。
DequeueCount	カウント	コンシューマーによって確認されたメッセージの数 (1 分あたり)。

メトリクス	単位	説明
InFlightCount	カウント	確認されていないコンシューマーに送信されたメッセージの数。
ReceiveCount	カウント	二重ネットワークコネクターに対してリモートブローカーから受信したメッセージの数。
MemoryUsage	割合 (%)	送信先が現在使用しているメモリ制限の割合。
ProducerCount	カウント	宛先のプロデューサーの数。
QueueSize	カウント	キュー内のメッセージの数。 **Important** このメトリクスは、キューにのみ適用されます。
TotalEnqueueCount	カウント	ブローカーに送信されたメッセージの合計数。
TotalDequeueCount	カウント	クライアントによって消費されたメッセージの合計数。

Note

TotalEnqueueCount および TotalDequeueCount メトリクスには、アドバイザリトピックのメッセージが含まれます。アドバイザリトピックメッセージの詳細については、ActiveMQ のドキュメントを参照してください。

ActiveMQ の送信先 (キューとトピック) メトリクスのディメンション

ディメンション	説明
Broker	ブローカーの名前。 **Note** 単一インスタンスブローカーにはサフィックス -1 があります。高可用性対応アクティブ/スタンバイブローカーには、冗長なペアに対してサフィックス -1 および -2 があります。
Topic、または Queue	トピックまたはキューの名前。
NetworkConnector	ネットワークコネクタの名前。

Amazon MQ for RabbitMQ ブローカーで利用可能な CloudWatch メトリクス

RabbitMQ ブローカーメトリクス

メトリクス	単位	説明
ExchangeCount	カウント	ブローカーで設定されたエクスチェンジの合計数。
QueueCount	カウント	ブローカーで設定されたキューの合計数。
ConnectionCount	カウント	ブローカーで確立された接続の合計数。
ChannelCount	カウント	ブローカーで確立されたチャネルの合計数。

メトリクス	単位	説明
ConsumerCount	カウント	ブローカーに接続されたコンシューマーの合計数。
MessageCount	カウント	キュー内のメッセージの合計数。 **Note** 生成される数値は、ブローカー上にある準備完了および未承認のメッセージの合計数です。
MessageReadyCount	カウント	キュー内の準備完了メッセージの合計数。
MessageUnacknowledgedCount	カウント	キュー内の未承認メッセージの合計数。
PublishRate	カウント	メッセージがブローカーに発行される速度。 生成される数値は、サンプリング時における 1 秒あたりのメッセージ数を表します。

メトリクス	単位	説明
ConfirmRate	カウント	RabbitMQ サーバーが発行されたメッセージを確認する速度。このメトリクスを PublishRate を比較して、ブローカーのパフォーマンスをより良く理解することができます。 生成される数値は、サンプリング時における 1 秒あたりのメッセージ数を表します。
AckRate	カウント	メッセージがコンシューマーによって承認される速度。 生成される数値は、サンプリング時における 1 秒あたりのメッセージ数を表します。
SystemCpuUtilization	割合 (%)	割り当てられた Amazon EC2 コンピュートユニット (ECU) のうち、現在ブローカーが使用しているユニットの割合。クラスターデプロイの場合、この値は 3 つの各 RabbitMQ ノードに対応するメトリクス値の集計を表します。
RabbitMQMemLimit	バイト	RabbitMQ ブローカーに対する RAM 制限。クラスターデプロイの場合、この値は 3 つの各 RabbitMQ ノードに対応するメトリクス値の集計を表します。

メトリクス	単位	説明
RabbitMQMemUsed	バイト	RabbitMQ ブローカーによって使用される RAM の量。クラスターデプロイの場合、この値は 3 つの各 RabbitMQ ノードに対応するメトリクス値の集計を表します。
RabbitMQDiskFreeLimit	バイト	RabbitMQ ブローカーに対するディスク制限。クラスターデプロイの場合、この値は 3 つの各 RabbitMQ ノードに対応するメトリクス値の集計を表します。このメトリクスは、インスタンスサイズごとに異なります。
RabbitMQDiskFree	バイト	RabbitMQ ブローカーで利用できる空きディスク領域の合計容量。ディスクの使用量が上限を超えると、クラスターはすべてのプロデューサー接続をブロックします。クラスターデプロイの場合、この値は 3 つの各 RabbitMQ ノードに対応するメトリクス値の集計を表します。
RabbitMQFdUsed	カウント	使用されたファイルディスクリプタの数。クラスターデプロイの場合、この値は 3 つの各 RabbitMQ ノードに対応するメトリクス値の集計を表します。

メトリクス	単位	説明
RabbitMQIOReadAverageTime	カウント	RabbitMQ が 1 回の読み込みオペレーションを実行する平均時間 (ミリ秒単位)。値はメッセージサイズに比例します。
RabbitMQIOWriteAverageTime	カウント	RabbitMQ が 1 回の書き込みオペレーションを実行する平均時間 (ミリ秒単位)。値はメッセージサイズに比例します。

RabbitMQ ブローカーメトリクスのディメンション

ディメンション	説明
Broker	ブローカーの名前。

RabbitMQ ノードメトリクス

メトリクス	単位	説明
SystemCpuUtilization	割合 (%)	割り当てられた Amazon EC2 コンピュートユニット (ECU) のうち、現在ブローカーが使用しているユニットの割合。
RabbitMQMemLimit	バイト	RabbitMQ ノードに対する RAM 制限。
RabbitMQMemUsed	バイト	RabbitMQ ノードによって使用される RAM の容量。メモリの使用量が制限を超えると、クラスターはすべてのプ

メトリクス	単位	説明
		ロデューサー接続をブロックします。
RabbitMQDiskFreeLimit	バイト	RabbitMQ ノードのディスク制限。このメトリクスは、インスタンスサイズごとに異なります。
RabbitMQDiskFree	バイト	RabbitMQ ノードで利用できる空きディスク領域の合計容量。ディスクの使用量が上限を超えると、クラスターはすべてのプロデューサー接続をブロックします。
RabbitMQFdUsed	カウント	使用されたファイルディスクリプタの数。

RabbitMQ ノードメトリクスのディメンション

ディメンション	説明
Node	ノードの名前。 **Note** ノード名は、プレフィックス (通常 rabbit) とホスト名の 2 つの部分で構成されます。例えば、rabbit@ip-10-0-0-230.us-west-2.compute.internal はプレフィックス rabbit とホスト名 ip-10-0-0-230.us-west-2.com

ディメンション	説明
	pute.internal を持つノード名です。
Broker	ブローカーの名前。

RabbitMQ キューメトリクス

メトリクス	単位	説明
ConsumerCount	カウント	キューにサブスクライブしているコンシューマーの数。
MessageReadyCount	カウント	現在配信可能なメッセージの数。
MessageUnacknowledgedCount	カウント	サーバーが承認を待機しているメッセージの数。
MessageCount	カウント	MessageReadyCount と MessageUnacknowledgedCount の合計数 (キュー深度とも呼ばれます)。

RabbitMQ キューメトリクスのディメンション

> **Note**
>
> Amazon MQ for RabbitMQ は、空白、タブ、またはその他の非 ASCII 文字が含まれた名前を持つ仮想ホストおよびキューのメトリクスを発行しません。
> ディメンション名の詳細については、Amazon CloudWatch API リファレンスの「Dimension」を参照してください。

ディメンション	説明
Queue	キューの名前。
VirtualHost	仮想ホストの名前。
Broker	ブローカーの名前。

RabbitMQ ネットワークメトリクス

メトリクス	単位	説明
NetworkOut	バイト	すべてのネットワークインターフェイスを通じ、このインスタンスから送信されたバイトの数。このメトリクスは 1 つのインスタンスからの送信ネットワークトラフィックの量を表しています。報告された数は期間中に送信されたバイト数です。基本 (5 分) のモニタリングで統計情報に 合計 使用している場合であれば、この数を 300 で除算してバイト/秒の値を求めることができます。詳細 (1 分) のモニタリングで統計情報に 合計 使用している場合合はこの数を 60 で除算します。CloudWatch メトリクスの計算関数 DIFF_TIME を使用して、1 秒あたりのバイト数を求めることもできます。例えば、CloudWatch で NetworkOut のグラフを m1 として作成した場合、Metric Math 数式 m1/(DIFF_TIME(m1)) はメトリクスをバイト/秒単位で返します。DIFF_TIME とその他の Metric Math 関数の詳細については、「Metric Math を使用する」を参照してください。 有意義な統計: 合計、平均、最小、最大
NetworkIn	バイト	すべてのネットワークインターフェイスを通じ、このインスタンスによって受信されたバイトの数。このメトリクスは 1 つのインスタンスへの受信ネットワークトラフィックの量を表しています。報告された数は期間中に受信されたバイト数です。基本 (5 分) のモニタリングで統計情報に 合

メトリクス	単位	説明
		計 使用している場合であれば、この数を 300 で除算してバイト/秒の値を求めることができます。詳細 (1 分) のモニタリングで統計情報に 合計 使用している場合合はこの数を 60 で除算します。CloudWatch メトリクスの計算関数 `DIFF_TIME` を使用して、1 秒あたりのバイト数を求めることもできます。例えば、CloudWatch で NetworkIn のグラフを `m1` として作成した場合、Metric Math 数式 `m1/(DIFF_TIME(m1))` はメトリクスをバイト/秒単位で返します。`DIFF_TIME` とその他の Metric Math 関数の詳細については、「Metric Math を使用する」を参照してください。 有意義な統計: 合計、平均、最小、最大

RabbitMQ ブローカーのディメンション

ディメンション	説明
`BrokerId`	ブローカーの ID

Amazon MQ for RabbitMQ ログの設定

RabbitMQ ブローカーに対して CloudWatch ロギングを有効にすると、Amazon MQ はサービスリンクロールを使用して CloudWatch に一般ログを発行します。ブローカーを初めて作成するときに Amazon MQ サービスリンクロールが存在しない場合、Amazon MQ がそのロールを自動的に作成します。すべての後続 RabbitMQ ブローカーは、同じサービスリンクロールを使用して CloudWatch にログを発行します。

サービスリンクロールの詳細については、「AWS Identity and Access Management ユーザーガイド」の「サービスリンクロールの使用」を参照してください。Amazon MQ がサービスリンクロールを使用する方法の詳細については、「the section called “サービスリンクロールの使用”」を参照してください。

を使用した Amazon MQ API コールのログ記録 AWS CloudTrail

Amazon MQ は、ユーザー AWS CloudTrail、ロール、またはサービスが行う Amazon MQ 呼び出しの記録を提供する AWS サービスである と統合されています。CloudTrail は、Amazon MQ ブローカーと設定に関連する API コールをイベントとしてキャプチャします。これには Amazon MQ コンソールからのコールと Amazon MQ API からのコードコールが含まれます。CloudTrail の詳細については、「AWS CloudTrail ユーザーガイド」を参照してください。

> **Note**
>
> CloudTrail は、ActiveMQ 操作 (メッセージの送受信など) や ActiveMQ ウェブコンソールに関連する API コールをログしません。ActiveMQ 操作に関連する情報をログするには、一般ログと監査ログを Amazon CloudWatch Logs に発行するように Amazon MQ を設定することができます。

CloudTrail が収集する情報を使用して、Amazon MQ API に対する特定のリクエスト、リクエスタの IP アドレス、リクエスタのアイデンティティ、およびリクエストの日時などを特定することができます。追跡を設定する場合は、Amazon S3 バケットへの CloudTrail イベントの継続的な配信を有効にすることができます 追跡を設定しない場合でも、CloudTrailコンソールのイベント履歴で最近のイベントを表示できます。詳細については、AWS CloudTrail ユーザーガイドの「Overview for Creating a Trail」を参照してください。

CloudTrail 内の Amazon MQ 情報

AWS アカウントを作成すると、CloudTrail が有効になります。サポートされている Amazon MQ イベントアクティビティが発生すると、イベント履歴内の他の AWS サービスイベントとともに CloudTrail イベントに記録されます。 AWS アカウントの最近のイベントを、表示、検索、およびダウンロードすることができます。詳細については、AWS CloudTrail ユーザーガイドの「CloudTrail イベント履歴でのイベントの表示」を参照してください。

追跡は、CloudTrailがログファイルを Amazon S3バケットに配信できるようにします。証跡を作成して、 AWS アカウントのイベントの継続的な記録を保持できます。デフォルトでは、 を使用して証跡を作成すると AWS マネジメントコンソール、証跡はすべての AWS リージョンに適用されます。証跡は、すべての AWS リージョンからのイベントを ログに記録し、ログファイルを指定された Amazon S3 バケットに配信します。CloudTrail ログで収集されたイベントデータをさらに分析して処理するように、他の AWS サービスを設定することもできます。詳細については、AWS CloudTrail ユーザーガイドの次のトピックを参照してください。

- CloudTrail がサポートするサービスと統合
- CloudTrail の Amazon SNS 通知の設定
- CloudTrail ログファイルの複数のリージョンからの受け取り
- 複数のアカウントから CloudTrailログファイルを受け取る

Amazon MQ は、以下の API のリクエストパラメータとレスポンスの両方を、イベントとして CloudTrail ログファイルにログすることをサポートします。

- `CreateConfiguration`
- `DeleteBroker`
- `DeleteUser`
- `RebootBroker`
- `UpdateBroker`

> **Note**
>
> RebootBroker ログファイルは、ブローカーを再起動したときに記録されます。メンテナンス期間中、サービスは自動的に再起動し、RebootBroker ログファイルは記録されません。

> **Important**
>
> 以下 API の GET メソッドの場合、リクエストパラメータはログ記録されますが、レスポンスは加工されます。
>
> - `DescribeBroker`
> - `DescribeConfiguration`
> - `DescribeConfigurationRevision`
> - `DescribeUser`
> - `ListBrokers`
> - `ListConfigurationRevisions`
> - `ListConfigurations`
> - `ListUsers`

以下の API では、data と password のリクエストパラメータはアスタリスク (***) によって非表示になります。

- CreateBroker (POST)
- CreateUser (POST)
- UpdateConfiguration (PUT)
- UpdateUser (PUT)

各イベントまたはログエントリには、リクエスタに関する情報が含まれます。この情報は以下のことを確認するのに役立ちます:

- リクエストが、ルートとユーザー認証情報のどちらを使用して送信されたか。
- リクエストが、ロールとフェデレーティッドユーザーのどちらの一時的なセキュリティ認証情報を使用して送信されたか。
- リクエストは別の AWS サービスによって行われましたか?

詳細については、「AWS CloudTrail ユーザーガイド」の「CloudTrail userIdentity Element」を参照してください。

Amazon MQ ログファイルエントリの例

追跡は、イベントを ログファイルとして指定された Amazon S3 バケットに配信することを可能にする設定です。CloudTrail のログファイルには、単一か複数のログエントリがあります。

イベントは、任意のソースからの単一のリクエストを表し、Amazon MQ API へのリクエスト、リクエスタの IP アドレス、リクエスタのアイデンティティ、およびリクエストの日時などに関する情報が含まれます。

以下の例は、CreateBroker API コールの CloudTrail ログエントリを示しています。

> **Note**
>
> CloudTrail ログファイルはパブリック API の順序付けられたスタックトレースではないため、特定の順序で情報が表示されることはありません。

```
{
    "eventVersion": "1.06",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AKIAIOSFODNN7EXAMPLE",
        "arn": "arn:aws:iam::111122223333:user/AmazonMqConsole",
        "accountId": "111122223333",
        "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
        "userName": "AmazonMqConsole"
    },
    "eventTime": "2018-06-28T22:23:46Z",
    "eventSource": "amazonmq.amazonaws.com",
    "eventName": "CreateBroker",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "203.0.113.0",
    "userAgent": "PostmanRuntime/7.1.5",
    "requestParameters": {
        "engineVersion": "5.15.9",
        "deploymentMode": "ACTIVE_STANDBY_MULTI_AZ",
        "maintenanceWindowStartTime": {
            "dayOfWeek": "THURSDAY",
            "timeOfDay": "22:45",
            "timeZone": "America/Los_Angeles"
        },
        "engineType": "ActiveMQ",
        "hostInstanceType": "mq.m5.large",
        "users": [
            {
                "username": "MyUsername123",
                "password": "***",
                "consoleAccess": true,
                "groups": [
                    "admins",
                    "support"
                ]
            },
            {
                "username": "MyUsername456",
                "password": "***",
                "groups": [
                    "admins"
                ]
            }
```

```
        ],
        "creatorRequestId": "1",
        "publiclyAccessible": true,
        "securityGroups": [
            "sg-a1b234cd"
        ],
        "brokerName": "MyBroker",
        "autoMinorVersionUpgrade": false,
        "subnetIds": [
            "subnet-12a3b45c",
            "subnet-67d8e90f"
        ]
    },
    "responseElements": {
        "brokerId": "b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9",
        "brokerArn": "arn:aws:mq:us-
east-2:123456789012:broker:MyBroker:b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9"
    },
    "requestID": "a1b2c345-6d78-90e1-f2g3-4hi56jk7l890",
    "eventID": "a12bcd3e-fg45-67h8-ij90-12k34d5l16mn",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```


Amazon MQ for ActiveMQ ログの設定

CloudWatch Logs へのログの発行を Amazon MQ に許可するには、ブローカーを作成および再起動する前に、Amazon MQ ユーザーに許可を追加するとともに、Amazon MQ のリソースベースポリシーも設定する必要があります。

> **Note**
>
> ActiveMQ ウェブコンソールからログを有効にしてメッセージを発行すると、メッセージのコンテンツが CloudWatch に送信され、ログに表示されます。

以下は、ActiveMQ ブローカー用の CloudWatch Logs を設定するステップの説明です。

トピック

- CloudWatch Logs でのロギングの構造を理解する

- Amazon MQ ユーザーへの CreateLogGroup 許可の追加
- Amazon MQ のリソースベースポリシーを設定する
- サービス間での不分別な代理処理の防止

CloudWatch Logs でのロギングの構造を理解する

一般ログおよび監査ログ記録は、高度なブローカー設定時、ブローカー作成時、またはブローカー編集時に有効にすることができます。

一般ロギングは、デフォルトの INFO ロギングレベルを有効にし (DEBUG ロギングはサポートされません)、activemq.log を CloudWatch アカウントのロググループに発行します。ロググループの形式は次のようになります。

```
/aws/amazonmq/broker/b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9/general
```

監査ロギングは、JMX または ActiveMQ ウェブコンソールを使用して行われた管理アクションのロギングを有効にし、audit.log を CloudWatch アカウントのロググループに発行します。ロググループの形式は次のようになります。

```
/aws/amazonmq/broker/b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9/audit
```

Amazon MQ は、単一インスタンスブローカーかアクティブ/スタンバイブローカーのどちらを使用しているかに応じて、各ロググループ内に 1 つまたは 2 つのログストリームを作成します。ログストリームの形式は次のようになります。

```
activemq-b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.log
                activemq-b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-2.log
```

サフィックスが -1 および -2 の場合は、個々のブローカーインスタンスを示します。詳細については、Amazon CloudWatch Logs ユーザーガイドの「ロググループとログストリームの操作」を参照してください。

Amazon MQ ユーザーへの **CreateLogGroup** 許可の追加

CloudWatch Logs ロググループの作成を Amazon MQ に許可するには、ブローカーを作成または再起動するユーザーに logs:CreateLogGroup アクセス許可があることを確認する必要があります。

> **Important**
>
> ユーザーがブローカーの作成または再起動を行う前に `CreateLogGroup` 許可をユーザーに追加しなければ、Amazon MQ はロググループを作成しません。

以下のサンプル IAM ベースポリシーは、このポリシーがアタッチされているユーザーの `logs:CreateLogGroup` に対する許可を付与します。

JSON

```
{
                    "Version":"2012-10-17",
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Action": "logs:CreateLogGroup",
                            "Resource": "arn:aws:logs:*:*:log-group:/aws/
amazonmq/*"
                        }
                    ]
                    }
```

> **Note**
>
> ここで、ユーザーという用語は、新しいブローカーの設定時に作成した Amazon MQ ユーザーではなく、ユーザーを指しています。ユーザーのセットアップと IAM ポリシーの設定の詳細については、IAM ユーザーガイドの「ID 管理の概要」セクションを参照してください。

詳細については、Amazon CloudWatch Logs API リファレンスの「`CreateLogGroup`」を参照してください。

Amazon MQ のリソースベースポリシーを設定する

> **Important**
>
> Amazon MQ にリソースベースポリシーを設定しない場合、ブローカーは CloudWatch Logs にログを発行できません。

CloudWatch Logs ロググループへのログの発行を Amazon MQ に許可するには、以下の CloudWatch Logs API アクションに対するアクセス権を Amazon MQ に付与するリソースベースポリシーを設定します。

- `CreateLogStream` – 指定したロググループの CloudWatch Logs ログストリームを作成します。
- `PutLogEvents` – 指定された CloudWatch Logs ログストリームにイベントを配信します。

次のリソースベースのポリシーは、 `logs:CreateLogStream`および へのアクセス許可を付与`logs:PutLogEvents`します AWS。

JSON

```
{
                        "Version":"2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Allow",
                                "Principal": { "Service":
 "mq.amazonaws.com" },
                                "Action": [ "logs:CreateLogStream",
 "logs:PutLogEvents" ],
                                "Resource": "arn:aws:logs:*:*:log-group:/aws/
amazonmq/*"
                            }
                        ]
                    }
```

このリソースベースのポリシーは、次のコマンド AWS CLI に示すように、 を使用して設定する必要があります。この例では、*us-east-1* を独自の情報に置き換えます。

```
aws --region us-east-1 logs put-resource-policy --policy-name AmazonMQ-logs \
                        --policy-document "{\"Version\": \"2012-10-17\", \"Statement\":
[{ \"Effect\": \"Allow\", \"Principal\": { \"Service\": \"mq.amazonaws.com\" },
                        \"Action\": [\"logs:CreateLogStream\", \"logs:PutLogEvents\"],
 \"Resource\": \"arn:aws:logs:*:*:log-group:\/aws\/amazonmq\/*\" }]}"
```

> **Note**
>
> この例では /aws/amazonmq/ プレフィックスを使用するため、リソースベースのポリシーは AWS アカウントごと、リージョンごとに 1 回のみ設定する必要があります。

サービス間での不分別な代理処理の防止

混乱した代理問題は、アクションを実行するためのアクセス許可を持たないエンティティが、より特権のあるエンティティにアクションの実行を強制できてしまう場合に生じる、セキュリティ上の問題です。では AWS、サービス間のなりすましにより、混乱した代理問題が発生する可能性があります。サービス間でのなりすましは、1 つのサービス (呼び出し元サービス) が、別のサービス (呼び出し対象サービス) を呼び出すときに発生する可能性があります。呼び出し元サービスは、本来ならアクセスすることが許可されるべきではない方法でその許可を使用して、別のお客様のリソースに対する処理を実行するように操作される場合があります。これを防ぐために、 は、アカウント内のリソースへのアクセス権が付与されたサービスプリンシパルを持つすべてのサービスのデータを保護するのに役立つツール AWS を提供します。

CloudWatch Logs アクセスを指定された 1 つまたは複数のブローカーに制限するには、Amazon MQ のリソースベースポリシーで `aws:SourceArn` および `aws:SourceAccount` のグローバル条件コンテキストキーを使用することをお勧めします。

> **Note**
>
> 両方のグローバル条件コンテキストキーを同じポリシーステートメントで使用する場合は、aws:SourceAccount 値と、aws:SourceArn 値に含まれるアカウントが、同じアカウント ID を示している必要があります。

次の例は、CloudWatch Logs アクセスを単一の Amazon MQ ブローカーに制限するリソースベースポリシーを示しています。

JSON

```
{
                        "Version":"2012-10-17",
                        "Statement": [
                            {
                            "Effect": "Allow",
                            "Principal": {
                                "Service": "mq.amazonaws.com"
                            },
                            "Action": [
                                "logs:CreateLogStream",
                                "logs:PutLogEvents"
                            ],
                            "Resource": "arn:aws:logs:*:*:log-group:/aws/
amazonmq/*",
                            "Condition": {
                                "StringEquals": {
                                "aws:SourceAccount": "123456789012",
                                "aws:SourceArn": "arn:aws:mq:us-
west-1:123456789012:broker:my-broker:123456789012"
                                }
                            }
                            }
                        ]
                        }
```

以下に示すように、CloudWatch Logs アクセスをアカウント内のすべてのブローカーに制限するように、リソースベースポリシーを設定することもできます。

JSON

```
{
                            "Version":"2012-10-17",
                            "Statement": [
                            {
                                "Effect": "Allow",
                                "Principal": {
                                "Service": [
                                    "mq.amazonaws.com"
```

```
                                ]
                                },
                                "Action": [
                                "logs:CreateLogStream",
                                "logs:PutLogEvents"
                                ],
                                "Resource": "arn:aws:logs:*:*:log-group:/aws/
amazonmq/*",
                                "Condition": {
                                "ArnLike": {
                                    "aws:SourceArn":
 "arn:aws:mq:*:123456789012:broker:*"
                                },
                                "StringEquals": {
                                    "aws:SourceAccount": "123456789012"
                                }
                                }
                            }
                            ]
                        }
```

「混乱した代理」セキュリティ問題の詳細については、ユーザーガイドの「混乱した代理問題」を参照してください。

Amazon MQ での CloudWatch Logs 設定のトラブルシューティング

場合によっては、CloudWatch Logs が常に期待通りに動作しないことがあります。このセクションでは、一般的な問題の概要とそれらの解決方法を説明します。

ロググループが CloudWatch に表示されない

`CreateLogGroup` 許可を Amazon MQ ユーザーに追加して、ブローカーを再起動します。そうすることで、Amazon MQ がロググループを作成できるようになります。

ログストリームが CloudWatch ロググループに表示されない

Amazon MQ のリソースベースポリシーを設定します。これにより、ブローカーよりログを発行することができます。

Amazon MQ のクォータ

このトピックでは、Amazon MQ の制限の一覧を示します。以下の制限の多くは、AWS のアカウント別に変更できます。制限緩和のリクエスト方法については、「Amazon Web Services 全般のリファレンス」の「AWS のサービスクォータ」を参照してください。上限の引き上げが適用された後でも、更新された上限は表示されません。Amazon CloudWatch での現在の接続上限の表示に関する詳細については、「Amazon CloudWatch を使用した Amazon MQ ブローカーのモニタリング」を参照してください。

トピック

ブローカー

以下の表は、Amazon MQ ブローカーに関連するクォータのリストです。

[制限]	説明
ブローカー名	• AWS アカウント内で一意にする必要があります。 • 1 ~ 50 文字にする必要があります。 • 使用できるのは、印刷可能な ASCII 文字に指定された文字のみです。 • 使用できるのは、英数字、ダッシュ、ピリオド、アンダースコア、チルダ (- . _ ~) のみです。
リージョンあたりのブローカー数	50

[制限]	説明
小規模ブローカーのプロトコルあたりのワイヤレベルの接続	**Important** RabbitMQ ブローカーには適用されません。 `mq.*.micro` インスタンスタイプのブローカーに対して 300 個。
大規模ブローカーのプロトコルあたりのワイヤレベルの接続	**Important** RabbitMQ ブローカーには適用されません。 `mq.*.*large` インスタンスタイプのブローカーに対して 2,000 個。
ブローカーあたりのセキュリティグループ	5
CloudWatch でモニタリングされる ActiveMQ 送信先 (キューとトピック)	CloudWatch は、最初の 1000 個の送信先のみをモニタリングします。
CloudWatch でモニタリングされる RabbitMQ 送信先 (キュー)	CloudWatch は、コンシューマーの数順に並べられた最初 500 個の送信先のみをモニタリングします。
ブローカーあたりのタグ	50

設定

以下の表は、Amazon MQ の設定に関連するクォータのリストです。

[制限]	説明
設定名	• 1 ~ 150 文字にする必要があります。 • 使用できるのは、印刷可能な ASCII 文字に指定された文字のみです。 • 使用できるのは、英数字、ダッシュ、ピリオド、アンダースコア、チルダ (- . _ ~) のみです。
設定あたりのリビジョン	300

Users

以下の表は、Amazon MQ ActiveMQ ブローカーのユーザーに関連するクォータのリストです。

[制限]	説明
ユーザーネーム	• 1 ~ 100 文字にする必要があります。 • 使用できるのは、印刷可能な ASCII 文字に指定された文字のみです。 • 使用できるのは、英数字、ダッシュ、ピリオド、アンダースコア、チルダ (- . _ ~) のみです。 • カンマ (,) を含めることはできません。
パスワード	• 12 ~ 250 文字にする必要があります。 • 使用できるのは、印刷可能な ASCII 文字に指定された文字のみです。 •

[制限]	説明
	少なくとも 4 個の一意文字を含める必要があります。 • カンマ (,) を含めることはできません。
ブローカーあたりのユーザー (simple auth)	250
ユーザーあたりのグループ (simple auth)	20

データストレージ

以下の表は、Amazon MQ のデータストレージに関連するクォータのリストです。

[制限]	説明
小規模なブローカーごとのストレージ容量	`mq.*.micro` インスタンスタイプのブローカーに対して 20 GB。Amazon MQ のインスタンスタイプの詳細については、「Broker instance types」を参照してください。
大規模なブローカーごとのストレージ容量	`mq.m5.*` インスタンスタイプのブローカーに対して 200 GB。Amazon MQ のインスタンスタイプの詳細については、「Broker instance types」を参照してください。
Amazon EBS によってバックアップされるブローカーごとのジョブスケジューラの使用制限	**Important** RabbitMQ ブローカーには適用されません。 50 GB。ジョブスケジューラの使用に関する詳細については、Apache ActiveMQ API ドキュメ

[制限]	説明
	ントの「JobSchedulerUsage 」を参照してください。
小規模なブローカーごとの一時的なストレージ容量	**Important** RabbitMQ ブローカーには適用されません。 mq.*.micro インスタンスタイプのブローカーに対して 5 GB。
大規模なブローカーごとの一時的なストレージ容量	**Important** RabbitMQ ブローカーには適用されません。 mq.m5.* インスタンスタイプのブローカーに対して 50 GB。

API スロットリング

以下のスロットリングクォータは、サービスの帯域幅を維持するために、すべての Amazon MQ API 全体で AWS アカウントごとに集計されます。Amazon MQ API の詳細については、Amazon MQ REST API リファレンスを参照してください。

Important

これらのクォータは、Amazon MQ for ActiveMQ または Amazon MQ for RabbitMQ のブローカーメッセージング API には適用されません。例えば、Amazon MQ はメッセージの送信または受信をスロットリングしません。

API バースト制限	API レート制限
100	15

Amazon MQ のトラブルシューティング

このセクションでは、Amazon MQ ブローカーの使用時に発生する可能性がある一般的な問題と、それらを解決するために実行できるステップについて説明します。一般的なトラブルシューティングヘルプについては、「the section called “トラブルシューティング: 一般的な Amazon MQ”」を参照してください。特定のエンジンバージョンのトラブルシューティングについては、以下のセクションを参照してください。

Amazon MQ の ActiveMQ のトラブルシューティング

トラブルシューティング情報	説明
一般的なトラブルシューティング	このセクションの情報を使用して、Amazon MQ の ActiveMQ ブローカーの使用時に発生する可能性がある一般的な問題の診断と解決に役立てます。
BROKER_ENI_DELETED	ブローカーの Elastic Network Interface (ENI) を削除すると、Amazon MQ の ActiveMQ は、BROKER_ENI_DELETED アラームを発生させます。
BROKER_OOM	Amazon MQ の ActiveMQ は、メモリ容量が不十分なためにブローカーが再起動ループの状態になると BROKER_OOM アラームを発生させます。

Amazon MQ の RabbitMQ のトラブルシューティング

トラブルシューティング情報	説明
一般的なトラブルシューティング	RabbitMQ ブローカーを操作するときに発生する可能性のある一般的な問題を診断します。

トラブルシューティング情報	説明
RABBITMQ_MEMORY_ALARM	RabbitMQ は、CloudWatch メトリクス `RabbitMQMemUsed` によって識別されるブローカーのメモリ使用量が、`RabbitMQMemLimit` によって識別されるメモリ制限を超えると、高メモリアラームを生成します。
RABBITMQ_INVALID_KMS_KEY	Amazon MQ の RabbitMQ は、カスタマーマネージド AWS KMS key(CMK) で作成されたブローカーが AWS Key Management Service (KMS) キーが無効であることを検出したときに、INVALID_KMS_KEY の重要なアクション必須コードを生成します。
RABBITMQ_INVALID_ASSUMEROLE	Amazon MQ の RabbitMQ は、 で指定された IAM ロール ARN が Amazon MQ によって引き受け`aws.arns.assume_role_arn` られない場合に、INVALID_ASSUMEROLE の重要なアクション必須コードを生成します。

トラブルシューティング情報	説明
RABBITMQ_INVALID_ARN_LDAP	Amazon MQ の RabbitMQ は、LDAP サービスアカウントのパスワード ARN が無効またはアクセスできない場合、INVALID_ARN_LDAP の重要なアクション必須コードを生成します。
RABBITMQ_INVALID_ARN_HTTP	Amazon MQ の RabbitMQ は、HTTP auth_backend の SSL 証明書またはキーファイルの 1 つ以上の ARNs が無効またはアクセスできない場合、INVALID_ARN_HTTP の重要なアクションに必要なコードを生成します。
RABBITMQ_INVALID_ARN_SSL	Amazon MQ の RabbitMQ は、EXTERNAL auth_mechanism の CA 証明書トラストストアの 1 つ以上の ARNs が無効またはアクセスできない場合、INVALID_ARN_SSL の重要なアクションに必要なコードを生成します。
RABBITMQ_INVALID_ARN	Amazon MQ の RabbitMQ は、ブローカー設定の 1 つ以上の ARNs が無効またはアクセスできない場合、INVALID_ARN の重要なアクション必須コードを生成します。

トラブルシューティング情報	説明
RABBITMQ_DISK_ALARM	ディスク制限アラームは、新しいメッセージが追加されるー方で消費されないメッセージが多いため、RabbitMQ ノードが使用するディスク量が減少したことを示します。

トラブルシューティング: 一般的な Amazon MQ

このセクションの情報を使用して、ブローカーへの接続問題、またはブローカーの再起動などの、Amazon MQ ブローカーの使用時に発生する可能性がある一般的な問題の診断に役立てます。

目次

ブローカーのウェブコンソールまたはエンドポイントに接続できません。

ウェブコンソールまたはワイヤレベルのエンドポイントを使用したブローカーへの接続で問題が発生する場合は、以下の手順が推奨されます。

1. ファイアウォールの内側からブローカーに接続しようとしているかどうかをチェックします。ブローカーへのアクセスを許可するようにファイアウォールを設定する必要がある場合があります。
2. FIPS エンドポイントを使用して、ブローカーに接続しようとしているかどうかをチェックしてください。Amazon MQ では、API オペレーションを使用する場合のみ FIPS エンドポイントがサポートされ、ブローカーインスタンス自体へのワイヤレベルの接続はサポートされません。
3. ブローカーの [Public Accessibility] (パブリックアクセシビリティ) オプションが [Yes] (はい) に設定されているかどうかをチェックします。これが [No] (いいえ) に設定されている場合は、サブネットのネットワークアクセスコントロールリスト (ACL) ルールをチェックしてください。カス

タムネットワーク ACL を作成した場合は、ブローカーへのアクセス権を提供するようにネットワーク ACL ルールを変更する必要がある場合があります。Amazon VPC ネットワークの詳細については、「Amazon VPC ユーザーガイド」の「インターネットアクセスを有効にする」を参照してください。

4. ブローカーのセキュリティグループルールをチェックします。以下のポートへの接続が許可されていることを確認してください。

> **Note**
>
> Amazon MQ の ActiveMQ と Amazon MQ の RabbitMQ は接続に異なるポートを使用するため、以下のポートはエンジンタイプ別に分類されています。

Amazon MQ の ActiveMQ

- ウェブコンソール – ポート 8162
- OpenWire – ポート 61617
- AMQP – ポート 5671
- STOMP — ポート 61614
- MQTT – ポート 8883
- WSS – ポート 61619

Amazon MQ の RabbitMQ

- ウェブコンソールおよび Management API – ポート　443 および 15671
- AMQP – ポート 5671

5. ブローカーエンジンタイプに対して、以下のネットワーク接続テストを実行します。

> **Note**
>
> パブリックアクセシビリティがないブローカーの場合は、Amazon MQ ブローカーと同じ Amazon VPC 内の Amazon EC2 インスタンスからテストを実行して、レスポンスを評価してください。

ActiveMQ on Amazon MQ

Amazon MQ の ActiveMQ ブローカーのネットワーク接続をテストする

1. 新規のターミナルまたはコマンドラインウィンドウを開きます。
2. 以下の nslookup コマンドを実行して、ブローカー DNS レコードをクエリします。アクティブ/スタンバイデプロイの場合は、アクティブエンドポイントとスタンバイエンドポイントの両方をテストします。アクティブ/スタンバイエンドポイントは、一意のブローカー ID に追加された -1 または -2 サフィックスで特定されます。エンドポイントを独自の情報に置き換えます。

```
$ nslookup b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-west-2.amazonaws.com
```

クエリが正常に完了すると、以下のような出力が表示されます。

```
Non-authoritative answer:
Server:  dns-resolver-corp-sfo-1.sfo.corp.amazon.com
Address:  172.10.123.456

Name:    ec2-12-345-123-45.us-west-2.compute.amazonaws.com
Address:  12.345.123.45
Aliases:  b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-west-2.amazonaws.com
```

解決された IP アドレスが、Amazon MQ コンソールで指定した IP アドレスと一致している必要があります。これは、ドメイン名が DNS サーバーで正しく解決されていることを示すので、次のステップに進むことができます。

3. 以下の telnet コマンドを実行して、ブローカーのネットワークパスをテストします。エンドポイントを独自の情報に置き換えます。必要に応じて、*port* をウェブコンソールのポート番号 8162、またはその他のワイヤレベルのポートに置き換えて、追加のプロトコルをテストします。

> **Note**
>
> アクティブ/スタンバイデプロイの場合、スタンバイエンドポイントで telnet を実行すると、Connect failed エラーメッセージが返されます。スタンバイインスタンス自体は実行されていますが、ActiveMQ プロセスは実行されておらず、ブローカーの Amazon EFS ストレージボリュームへのアクセス権がないため、これ

は期待どおりの動作です。アクティブインスタンスとスタンバイインスタンスの両方をテストできるように、-1 および -2 両方のエンドポイントにこのコマンドを実行してください。

```
$ telnet b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-
west-2.amazonaws.com port
```

アクティブインスタンスには、以下のような出力が表示されます。

```
Connected to b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-
west-2.amazonaws.com.
Escape character is '^]'.
```

4. 以下のいずれかを行ってください。

   - telnet コマンドが正常に完了する場合は、EstablishedConnectionsCount メトリクスをチェックして、ブローカーがワイヤレベル接続の上限に到達していないことを確認します。ブローカーの General ログを調べて、上限に到達したかどうかを確認することも可能です。このメトリクスがゼロより大きい場合は、現在少なくとも 1 つのクライアントがブローカーに接続されています。メトリクスがゼロ個の接続を示している場合は、telnet パステストを再度実行し、少なくとも 1 分待ってから接続を切断してください (ブローカーメトリクスは毎分発行されるため)。
   - telnet コマンドが失敗する場合は、ブローカーの Elastic Network Interface のステータスをチェックして、ステータスが in-use になっていることを確認します。各インスタンスのネットワークインターフェイスに関する Amazon VPC フローログを作成して、生成されたフローログを検証します。telnet コマンドを実行したときのブローカーの IP アドレスを調べて、応答パケットを含む接続パケットが ACCEPTED であることを確認します。フローログの詳細と例については、Amazon VPC デベロッパーガイドの「フローログレコードの例」を参照してください。

5. 以下の curl コマンドを実行して、ActiveMQ の管理ウェブコンソールへの接続をチェックします。

```
$ curl https://b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-
west-2.amazonaws.com:8162/index.html
```

コマンドが正常に完了すると、出力は以下のような HTML ドキュメントになります。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;
 charset=iso-8859-1" />
        <title>Apache ActiveMQ</title>
        ...
```

RabbitMQ on Amazon MQ

Amazon MQ の RabbitMQ ブローカーのネットワーク接続をテストする

1. 新規のターミナルまたはコマンドラインウィンドウを開きます。
2. 以下の nslookup コマンドを実行して、ブローカー DNS レコードをクエリします。エンドポイントを独自の情報に置き換えます。

```
$ nslookup b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-west-2.amazonaws.com
```

クエリが正常に完了すると、以下のような出力が表示されます。

```
Non-authoritative answer:
Server:  dns-resolver-corp-sfo-1.sfo.corp.amazon.com
Address:  172.10.123.456

Name:    rabbit-broker-1c23e456ca78-b9000123b4ebbab5.elb.us-
west-2.amazonaws.com
Addresses:  52.12.345.678
          52.23.234.56
          41.234.567.890
          54.123.45.678
Aliases:  b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-west-2.amazonaws.com
```

3. 以下の telnet コマンドを実行して、ブローカーのネットワークパスをテストします。エンドポイントを独自の情報に置き換えます。*port* をウェブコンソールのポート 443 に置き換える、および 5671 に置き換えてワイヤレベルの AMQP 接続をテストすることができます。

```
$ telnet b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-
west-2.amazonaws.com port
```

コマンドが正常に完了する場合は、以下のような出力が表示されます。

```
Connected to b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-
west-2.amazonaws.com.
Escape character is '^]'.
```

> **Note**
>
> Telnet 接続は、数秒後に自動的に終了します。

4. 以下のいずれかを行ってください。

- telnet コマンドが正常に完了する場合は、ConnectionCount メトリクスをチェックして、max-connections デフォルトポリシーで設定されている値にブローカーが到達していないことを確認します。ブローカーの Connection.log ロググループを調べて、上限に到達したかどうかを確認することも可能です。このメトリクスがゼロより大きい場合は、現在少なくとも 1 つのクライアントがブローカーに接続されています。メトリクスがゼロ個の接続を示している場合は、telnet パステストを再度実行します。ブローカーが新しい接続メトリクスを CloudWatch に発行する前に接続が終了する場合は、このプロセスを繰り返す必要がある場合があります。メトリクスは毎分発行されます。
- パブリックアクセシビリティがないブローカーで telnet コマンドが失敗する場合は、ブローカーの Elastic Network Interface のステータスをチェックして、ステータスが in-use になっていることを確認します。各ネットワークインターフェイスに関する Amazon VPC フローログを作成して、生成されたフローログを検証します。telnet コマンドが呼び出されたときのブローカーのプライベート IP アドレスを調べて、応答パケットを含む接続パケットが ACCEPTED であることを確認します。フローログの詳細と例については、Amazon VPC デベロッパーガイドの「フローログレコードの例」を参照してください。

> **Note**
>
> このステップは、パブリックアクセシビリティがある Amazon MQ の RabbitMQ ブローカーには適用されません。

5. 以下の `curl` コマンドを実行して、RabbitMQ の管理ウェブコンソールへの接続をチェックします。

```
$ curl https://b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-west-2.amazonaws.com:443/index.html
```

コマンドが正常に完了すると、出力は以下のような HTML ドキュメントになります。

```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="X-UA-Compatible" content="IE=edge" />
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>RabbitMQ Management</title>
        ...
```

ブローカーが実行中であり、**telnet** を使用して接続を検証できますが、クライアントは接続できず、SSL 例外を返しています。

ブローカーのエンドポイント証明書がブローカーのメンテナンスウィンドウ中に更新されている可能性があります。Amazon MQ ブローカー証明書は定期的にローテーションされ、ブローカーの可用性とセキュリティが引き続き維持されます。

Amazon Trust Services で Amazon ルート認証局 (CA) を使用して、クライアントのトラストストアで認証することをお勧めします。すべての Amazon MQ ブローカーの証明書は、このルート CA で署名されています。Amazon ルート CA を使用することで、ブローカーで証明書が更新されるたびに、新しい Amazon MQ ブローカーの証明書をダウンロードする必要がなくなります。

ブローカーを作成しましたが、ブローカーの作成に失敗しました。

ブローカーが `CREATION_FAILED` ステータスになっている場合は、以下の手順を実行します。

- IAM 許可をチェックします。ブローカーを作成するには、 AWS マネージド IAM ポリシーを使用するか、カスタム IAM ポリシーに正しい Amazon EC2 アクセス許可のセットAmazonMQFullAccessが必要です。必要な Amazon EC2 許可の詳細については、「Amazon MQ ブローカーの作成に必要な IAM 許可」を参照してください。
- ブローカー用に選択しているサブネットが、共有 Amazon Virtual Private Cloud (VPC) 内にあるかどうかをチェックします。共有 Amazon VPC 内で Amazon MQ ブローカーを作成するには、その Amazon VPC を所有するアカウントでブローカーを作成する必要があります。

ブローカーが再起動したのですが、その理由がよくわかりません。

ブローカーが自動的に再起動した場合は、以下の理由のいずれかが原因で再起動した可能性があります。

- スケジュールされた毎週のメンテナンスウィンドウが原因でブローカーが再起動した可能性があります。Amazon MQ は、ハードウェア、オペレーティングシステム、またはメッセージブローカーのエンジンソフトウェアに対して定期的にメンテナンスを実行します。メンテナンスの所要時間はさまざまですが、メッセージブローカーに対してスケジュールされている操作によっては、最長 2 時間継続することがあります。ブローカーは、この 2 時間のメンテナンスウィンドウのどの時点でも再起動する可能性があります。ブローカーのメンテナンスウィンドウの詳細については、「the section called “ブローカーのメンテナンスのスケジュール”」を参照してください。
- ブローカーのインスタンスタイプがアプリケーションワークロードに適していない可能性があります。例えば、mq.t3.micro で実稼働ワークロードを実行すると、ブローカーのリソースが不足する原因になる場合があります。CPU 使用率が高い、またはブローカーのメモリ使用率が高いと、ブローカーが予期せず再起動する原因になる場合があります。ブローカーが使用する CPU とメモリの量を確認するには、エンジンタイプに対応する以下の CloudWatch メトリクスを使用してください。
  - Amazon MQ の ActiveMQ – 割り当てられた Amazon EC2 コンピューティングユニットのうち、ブローカーが現在使用している割合について CpuUtilization をチェックします。ActiveMQ JVM メモリ制限のうち、ブローカーが現在使用している割合について HeapUsage をチェックします。
  - Amazon MQ の RabbitMQ – 割り当てられた Amazon EC2 コンピューティングユニットのうち、ブローカーが現在使用している割合について SystemCpuUtilization をチェックします。使用済みの RAM の量 (バイト単位) について RabbitMQMemUsed をチェックし、それを RabbitMQMemLimit で除算して、RabbitMQ ノードが使用したメモリの割合を算出します。

ブローカーのインスタンスタイプ、およびワークロードに適したインスタンスタイプを選択する方法の詳細については、「Broker instance types」を参照してください。

Amazon MQ の ActiveMQ のトラブルシューティング

このセクションの情報を使用して、Amazon MQ の ActiveMQ ブローカーの使用時に発生する可能性がある一般的な問題の診断と解決に役立てます。

目次

ロギングをアクティブにしているにもかかわらず、CloudWatch Logs にブローカーの一般ログまたは監査ログが表示されません。

CloudWatch Logs でブローカーのログを表示できない場合は、以下の手順を実行します。

1. ブローカーを作成または再起動するユーザーに `logs:CreateLogGroup` アクセス許可があるかどうかを確認します。ユーザーがブローカーの作成または再起動を行う前に `CreateLogGroup` 許可をユーザーに追加しなければ、Amazon MQ はロググループを作成しません。
2. Amazon MQ が CloudWatch Logs にログを発行することを許可するリソースベースポリシーが設定されているかどうかをチェックします。CloudWatch Logs ロググループへのログの発行を Amazon MQ に許可するには、以下の CloudWatch Logs API アクションに対するアクセス権を Amazon MQ に付与するリソースベースポリシーを設定します。
   - `CreateLogStream` – 指定したロググループの CloudWatch Logs ログストリームを作成します。
   - `PutLogEvents` – 指定された CloudWatch Logs ログストリームにイベントを配信します。

CloudWatch Logs にログを発行するための Amazon MQ の ActiveMQ の設定に関する詳細については、「ロギングの設定」を参照してください。

ブローカーの再起動またはメンテナンスウィンドウ後、ステータスが **RUNNING** であってもブローカーに接続できない。目的

開始したブローカーの再起動後、スケジュールされたメンテナンスウィンドウの完了後、またはスタンバイインスタンスがアクティブ化された障害イベントで、接続の問題が発生する可能性があります。いずれの場合も、ブローカーの再起動後の接続の問題は、ブローカーの Amazon EFS または Amazon EBS ストレージボリュームに保持されるメッセージが異常に多数であることが原因である可能性が最も高いです。再起動中、Amazon MQ は永続化されたメッセージをストレージからブローカメモリに移動します。この診断を確認するには、CloudWatch で Amazon MQ for ActiveMQ ブローカーの次のメトリクスを監視します。

- **StoragePercentUsage** — 100% に近づくほど割合が大きいと、ブローカーは接続を拒否する可能性があります。
- **JournalFilesForFullRecovery** — クリーンでないシャットダウンおよび再起動後に再生されるジャーナルファイルの数を示します。値が増加する、または常に高い値は、再起動後に接続の問題を引き起こす可能性のある未解決のトランザクションを示します。
- **OpenTransactionCount** — 再起動後にゼロより大きい数字は、ブローカーが以前に消費されたメッセージを保存しようとし、その結果、接続の問題が発生することを示します。

この問題を解決するには、XA トランザクションを `rollback()` または `commit()` で解決することをお勧めします。詳細および `rollback()` を使用して XA トランザクションを解決するコード例を確認するには、「XAトランザクションの回復」を参照してください。

一部のクライアントはブローカーに接続していますが、他のクライアントは接続できません。

ブローカーが `RUNNING` ステータスであり、一部のクライアントはブローカーに正常に接続できますが、他のクライアントは正常に接続できません。ブローカーのワイヤレベル接続の上限に達している可能性があります。ワイヤレベルの接続制限に達したことを確認するには、次の手順を実行します。

- CloudWatch Logs で Amazon MQ の ActiveMQ ブローカーの一般的なブローカーログを確認します。上限に達した場合は、ブローカーログに `Reached Maximum Connections` が表示されます。Amazon MQ の ActiveMQ ブローカーに対する CloudWatch Logs の設定に関する詳細につい

ては、「the section called “CloudWatch Logs での口ギングの構造を理解する”」を参照してください。

ワイヤレベルの接続制限に達すると、ブローカーは追加の着信接続を積極的に拒否します。この問題を解決するには、ブローカーインスタンスタイプをアップグレードすることをお勧めします。ワークロードに最適なインスタンスタイプの選択の詳細については、「Broker instance types」を参照してください。

ワイヤレベル接続の数がブローカー接続制限を下回っていることを確認できた場合、問題はクライアントの再起動に関連している可能性があります。ブローカーのログで、`... Inactive for longer than 600000 ms - removing ...` の多数で頻繁なエントリを確認してください。ログエントリは、クライアントの再起動または接続の問題を示しています。この影響は、頻繁にブローカーを切断して再接続するクライアントと Network Load Balancer (NLB) を介してブローカーに接続する場合に顕著になります。これは通常、コンテナベースのクライアントで観察されます。

詳細については、クライアント側のログを確認してください。ブローカーは 600000 ミリ秒後に非アクティブな TCP 接続をクリーンアップし、接続ソケットを解放します。

オペレーションを実行すると、ActiveMQ コンソールに例外 **org.apache.jasper.JasperException: An exception occurred processing JSP page** が表示されます。

簡易認証を使用していて、キューとトピックの認可に `AuthorizationPlugin` を設定している場合は、XML 設定ファイルで `AuthorizationEntries` 要素を使用し、`activemq-webconsole` グループにすべてのキューとトピックへのアクセス許可を付与してください。これにより、ActiveMQ ウェブコンソールが ActiveMQ ブローカーと通信できるようになります。

次のサンプルの `AuthorizationEntry` は、`activemq-webconsole` グループにすべてのキューとトピックの読み取りおよび書き込みの許可を付与します。

```
<authorizationEntries>
    <authorizationEntry admin="activemq-webconsole,admins,users" topic=">"
 read="activemq-webconsole,admins,users" write="activemq-webconsole,admins,users" />
    <authorizationEntry admin="activemq-webconsole,admins,users" queue=">"
 read="activemq-webconsole,admins,users" write="activemq-webconsole,admins,users" />
</authorizationEntries>
```

同様に、ブローカーを LDAP に統合する場合は、必ず `amazonmq-console-admins` グループに許可を付与してください。LDAP の詳細については、「the section called “LDAP 統合の仕組み”」を参照してください。

トラブルシューティング: Amazon MQ の RabbitMQ

このセクションの情報を使用して、Amazon MQ の RabbitMQ ブローカーの使用時に発生する可能性がある一般的な問題の診断と解決に役立てます。

目次

CloudWatch にキューまたは仮想ホストのメトリクスが表示されません。

CloudWatch にキューまたは仮想ホストのメトリクスが表示されない場合は、キューまたは仮想ホストの名前に、空白、タブ、またはその他の非 ASCII 文字が含まれていないか確認してください。

Amazon MQ は、空白、タブ、またはその他の非 ASCII 文字が含まれた名前を持つ仮想ホストおよびキューのメトリクスを発行できません。

ディメンション名の詳細については、「Amazon CloudWatch API リファレンス」の「Dimension」を参照してください。

Amazon MQ の RabbitMQ でプラグインを有効にするにはどうすればよいですか?

Amazon MQ の RabbitMQ は現在、デフォルトで有効になっている RabbitMQ 管理、シャベル、フェデレーション、コンシステントハッシュ交換プラグインのみをサポートしています。サポートされているプラグインの詳細については、「the section called “プラグイン”」を参照してください。

ブローカーの Amazon VPC 設定を変更できません。

Amazon MQ は、ブローカーが作成された後の Amazon VPC 設定の変更をサポートしていません。新しい Amazon VPC 設定で新しいブローカーを作成し、クライアント接続 URL を新しいブローカー接続 URL で更新する必要があることに注意してください。

クラスターのデプロイでキューの同期が一時停止しました。

RabbitMQ の高メモリアラームに対処しているときに、1 つまたは複数のキューのメッセージを消費できないことがあります。これらのキューは、ノード間でメッセージを同期中である可能性があります。その間、それぞれのキューは、メッセージの発行および消費に使用できなくなります。高メモリアラームが原因でキューの同期が一時停止し、メモリアラームの原因になることさえあります。

一時停止したキューの同期の停止と再試行の詳細については、「the section called “一時停止されたキュー同期の解決”」を参照してください。

Amazon MQ for RabbitMQ シングルインスタンスブローカーが再起動ループにあります。

高メモリアラームを発生させる Amazon MQ for RabbitMQ の単一インスタンスブローカーは、再起動時に起動するための十分なメモリがない場合、利用できなくなる可能性があります。これにより、RabbitMQ が再起動のループに入り、問題が解決するまでブローカーとのやり取りが妨げられる可能性があります。ブローカーが再起動ループにある場合、Amazon MQ が推奨するベストプラクティスを適用して高メモリアラームを解決することはできません。

ブローカーを回復させるには、より多くのメモリを持つ大きなインスタンスタイプにアップグレードすることをお勧めします。クラスターのデプロイとは異なり、再起動中にノード間で実行するキューの同期がないため、高メモリアラームの発生時に単一インスタンスブローカーをアップグレードできます。

ブローカーのすべての管理者アカウントへのアクセスを失いました。

IAM 認証を使用してアクセスを復旧できます。 AWS アカウントのアウトバウンドウェブ ID フェデレーションを有効にし、ウェブ ID トークンを取得するアクセス許可を持つ IAM ロールを作成し、OAuth 2.0 経由で IAM 認証を受け入れるようにブローカーを設定し、IAM 認証情報を使用して JWT トークンを取得し、新しい管理者ユーザーを作成します。詳細な手順については、「the section called “IAM 認証と認可の使用”」を参照してください。

Amazon MQ の ActiveMQ: 削除された Elastic Network Interface のアラーム

ブローカーの Elastic Network Interface (ENI) を削除すると、Amazon MQ の ActiveMQ は、BROKER_ENI_DELETED アラームを発生させます。初めて Amazon MQ ブローカーを作成するときは、Amazon MQ がアカウントの Virtual Private Cloud (VPC) 内に Elastic Network Interface をプロビジョンするため、多数の EC2 許可が必要になります。

このネットワークインターフェイスを変更または削除しないでください。このネットワークインターフェイスを変更または削除すると、VPC とブローカーとの間の接続が完全に失われる可能性があります。ネットワークインターフェイスを削除する場合は、まずブローカーを削除します。

Amazon MQ の ActiveMQ: ブローカーのメモリ不足アラーム

Amazon MQ の ActiveMQ は、メモリ容量が不十分なためにブローカーが再起動ループの状態になると BROKER_OOM アラームを発生させます。ブローカーが再起動ループ (バウンスループとも呼ばれる) の状態になると、ブローカーは短時間内にリカバリの試行を繰り返します。メモリ容量不足でスタートアップを完了できないブローカーは、再起動のループに入る可能性があり、その間はブローカーとのやり取りが制限されます。

Amazon MQ は、デフォルトでブローカーのメトリクスを有効にします。Amazon CloudWatch コンソールにアクセスするか、CloudWatch API を使用して、ブローカーのメトリクスを表示できます。次のメトリクスは、ActiveMQ BROKER_OOM アラームを診断する場合に役立ちます。

Amazon MQ CloudWatch メトリクス	メモリ使用量が多い理由	
`TotalMessageCount`	メッセージは、消費または破棄されるまでメモリに格納されます。メッセージ数が多いと、リソースの過剰使用が表示され、高メモリアラームの原因となる可能性があります。	
`HeapUsage`	ブローカーが現在使用している ActiveMQ JVM メモリ制限	

Amazon MQ CloudWatch メトリクス	メモリ使用量が多い理由	
	の割合。パーセンテージが高い場合は、ブローカーが大量のリソースを使用していることを示し、OOM アラームが発生する可能性があります。	
ConnectionCount	クライアント接続にはメモリを使用するため、同時接続が多すぎると高メモリアラームの原因となる可能性があります。	
CpuUtilization	割り当てられた EC2 コンピューティングユニットのうち、現在ブローカーが使用しているものの比率。	
TotalConsumerCount	ブローカーに接続されているすべてのコンシューマーについて、設定された数のメッセージは、コンシューマーに配信される前にストレージからメモリにロードされます。コンシューマーの接続が多いと、メモリ使用率が高くなり、高メモリアラームの原因となる可能性があります。	

再起動ループを防ぎ、BROKER_OOM アラームを回避するには、メッセージがすばやく消費されるようにします。これを行うには、最も効果的なブローカーインスタンスタイプを選択し、配信不能または期限切れのメッセージを破棄するために、デッドレターキューをクリーニングします。効果的なパフォーマンスを確保する方法の詳細については、「Amazon MQ の ActiveMQ のベストプラクティス」をご覧ください。

Amazon MQ for RabbitMQ: 高メモリアラーム

Amazon MQ for RabbitMQ は、CloudWatch メトリクス `RabbitMQMemUsed` によって識別されるブローカーのメモリ使用量が、`RabbitMQMemLimit` によって識別されるメモリ制限を超えると、高メモリアラームを生成します。

高メモリアラームが発生した RabbitMQ ブローカーでは、メッセージを発行しているすべてのクライアントがブロックされます。ブローカーが再起動ループに入ったり、キューの同期が一時停止されたり、アラームの診断と解決を複雑にするその他の問題が発生したりすることがあります。

高メモリアラームを診断して解決するには、まず RabbitMQ のすべてのベストプラクティスに従い、次のステップを実行します。

> **Important**
>
> - `RabbitMQMemLimit` は Amazon MQ によって設定され、ホストインスタンスタイプごとに使用可能なメモリを考慮して特別に調整されます。
> - Amazon MQ では、高メモリアラームが発生しているブローカーの再起動は行われません。また、ブローカーでアラームが発生し続ける限り `RebootBroker` API オペレーションに対して例外が返されます。

ステップ 1: 高メモリアラームを診断する

Amazon MQ for RabbitMQ ブローカーで高メモリアラームを診断するには、2 つの方法があります。CloudWatch で RabbitMQ ウェブコンソールと Amazon MQ メトリクスの両方をチェックすることをお勧めします。

RabbitMQ ウェブコンソールを使用した高メモリアラームの診断

RabbitMQ ウェブコンソールでは、各ノードのメモリ使用率の詳細情報を生成して表示できます。この情報は、次の手順を実行することで確認できます。

1. にサインイン AWS マネジメントコンソール し、ブローカーの RabbitMQ ウェブコンソールを開きます。
2. RabbitMQ コンソールの [Overview] (概要) ページで、[Nodes] (ノード) リストからノードの名前を選択します。

3. ノードの詳細ページで、[Memory details] (メモリの詳細) を選択してセクションを展開し、ノードにおけるメモリ使用率の情報を表示します。

RabbitMQ がウェブコンソールで提供するメモリ使用率の情報は、メモリを消費しすぎている可能性や、高メモリアラームの原因となる可能性のあるリソースを特定するのに役立ちます。RabbitMQ ウェブコンソールで使用できるメモリ使用率の詳細については、RabbitMQ Server Documentation ウェブサイトの「Reasoning About Memory Use」を参照してください。

Amazon MQ メトリクスを使用した高メモリアラームの診断

Amazon MQ は、デフォルトでブローカーのメトリクスを有効にします。CloudWatch コンソールにアクセスするか、CloudWatch API を使用して、ブローカーのメトリクスを表示できます。次のメトリクスは、RabbitMQ の高メモリアラームを診断する際に便利です。

Amazon MQ CloudWatch メトリクス	メモリ使用量が多い理由	
`MessageCount`	メッセージは、消費または破棄されるまでメモリに格納されます。メッセージ数が多いと、リソースの過剰使用が表示され、高メモリアラームの原因となる可能性があります。	
`QueueCount`	また、キューはメモリに格納されます。キューの数が多いと高メモリアラームの原因となる可能性があります。	
`ConnectionCount`	クライアント接続にはメモリを使用するため、同時接続が多すぎると高メモリアラームの原因となる可能性があります。	
`ChannelCount`	接続と同様に、各接続を使用して確立されたチャネルも	

Amazon MQ CloudWatch メトリクス	メモリ使用量が多い理由	
	ノードメモリに格納されます。チャネルの数が多いと高メモリアラームの原因となる可能性があります。	
`ConsumerCount`	ブローカーに接続されているすべてのコンシューマーについて、設定された数のメッセージは、コンシューマーに配信される前にストレージからメモリにロードされます。コンシューマーの接続が多いと、メモリ使用率が高くなり、高メモリアラームの原因となる可能性があります。	
`PublishRate`	メッセージの発行には、ブローカーのメモリが使用されます。メッセージがブローカーに発行される速度が高すぎて、ブローカーがコンシューマーにメッセージを配信する速度を大幅に上回ると、ブローカーで高メモリアラームが発生する可能性があります。	

ステップ 2: 高メモリアラームに対処して防止する

> **Note**
>
> 必要なアクションを実行した後、RABBITMQ_MEMORY_ALARM ステータスがクリアされるまでに数時間かかる場合があります。

一般的な防止方法として、RabbitMQ のすべてのベストプラクティスに従ってください。特定したコントリビューターごとに、RabbitMQ の高メモリアラームに防止して対処するため、次の一連のアクションをお勧めします。

メモリ使用量が多い原因	対応に関する Amazon MQ の推奨事項	防止のための Amazon MQ の推奨事項
メッセージの数	キューに発行されたメッセージの消費、キューからのメッセージの消去、またはブローカーからのキューの削除。	レイジーキューを有効にし、キューの深度制限を設定または削減します。
キューの数	キューの数を減らします。	キューの数の制限を設定するか、減らします。
接続の数	接続の数を減らします。	接続の数の制限を設定するか、減らします。
チャンネルの数	チャネルの数を減らします。	クライアントアプリケーションで、接続あたりのチャネルの最大数を設定します。
コンシューマー数	ブローカーに接続されたコンシューマーの数を減らします。	小さいコンシューマーのプリフェッチの制限を設定します。
メッセージの発行速度	パブリッシャーがメッセージをブローカーに発行する速度を低くします。	パブリッシャーの確認を有効にします。
クライアント接続試行レート	メッセージを発行または消費できるようにクライアントがブローカーへの接続を試行する頻度を減らすか、ブローカーを設定します。	より長時間の接続を使用して、接続の試行回数と頻度を減らします。

ブローカーのメモリアラームが解決したら、ホストインスタンスタイプを追加のリソースを含むインスタンスにアップグレードできます。ブローカーのインスタンスタイプを更新する方法については、「Amazon MQ REST API リファレンス」の「UpdateBrokerInput」を参照してください。

> **Note**
>
> ブローカーを mq.m5.x インスタンスタイプから mq.t3.micro インスタンスタイプにダウングレードすることはできません。ダウングレードするには、ブローカーを削除し、新しいブローカーを作成する必要があります。

Amazon MQ の RabbitMQ: 無効な AWS Key Management Service キー Amazon MQ

Amazon MQ の RabbitMQ は、カスタマーマネージド AWS KMS key(CMK) で作成されたブローカーが AWS Key Management Service (KMS) キーが無効であることを検出したときに、INVALID_KMS_KEY の重要なアクション必須コードを生成します。CMK を備えた RabbitMQ ブローカーは、KMS キーが有効になっていることと、ブローカーに必要な権限がすべて付与されていることを定期的に確認します。キーが有効になっていることを RabbitMQ が確認できない場合、ブローカーは隔離され、RabbitMQ は INVALID_KMS_KEY を返します。

有効な KMS キーがない場合、ブローカーにはカスタマー管理の KMS キーに対する基本的なアクセス許可がありません。ユーザーがキーを再度有効にしてブローカーが再起動するまで、ブローカーはキーを使用して暗号化操作を実行できません。KMS キーが無効になっている RabbitMQ ブローカーは、劣化を防ぐために隔離されます。KMS キーが再び有効になったことを RabbitMQ が確認すると、ブローカーは隔離から除外されます。Amazon MQ は、KMS キーが無効になっているブローカーを再起動せず、ブローカーが無効な KMS キーを保持し続ける限り、RebootBroker API オペレーションに対して例外を返します。

INVALID_KMS_KEY の診断と対処

INVALID_KMS_KEY アクションに必要なコードを診断して対処するには、 コマンドラインインターフェイス (CLI) AWS と AWS Key Management Service コンソールを使用する必要があります。

KMS キーを再度有効にするには

1. DescribeBroker メソッドを呼び出して CMK ブローカーの kmsKeyId を取得します。
2. AWS Key Management Service コンソールにサインインします。

3. [カスタマー管理キー] ページで、問題のあるブローカーの KMS キー ID を見つけて、ステータスが [有効] であることを確認します。
4. KMS キーが無効になっている場合は、[キーアクション]、[有効化] の順に選択してキーを再度有効にします。キーを再度有効にしたら、RabbitMQ がブローカーを隔離から除外するまで待つ必要があります。

必要な許可がまだブローカーの KMS キーに関連付けられていることを確認するには、`ListGrantListGrant` メソッドを呼び出して、`mq_rabbit_grant` と `mq_grant` が存在することを確認します。KMS 許可またはキーが削除されている場合は、ブローカーを削除し、必要な許可をすべて備えた新しいブローカーを作成する必要があります。ブローカーを削除する手順については、「ブローカーの削除」を参照してください。

重要なアクションが必要なコード INVALID_KMS_KEY が発生しないようにするには、KMS キーまたは CMK 許可を手動で削除または無効化しないでください。キーを削除する場合は、まずブローカーを削除します。

Amazon MQ の RabbitMQ: ディスク制限アラーム

ディスク制限アラームは、新しいメッセージが追加される一方で消費されないメッセージが多いため、RabbitMQ ノードが使用するディスク量が減少したことを示します。RabbitMQ は、Amazon CloudWatch メトリクス `RabbitMQDiskFree` で識別されるブローカーの空きディスク容量が、`RabbitMQDiskFreeLimit` で識別されるディスク制限に達すると、ディスク制限アラームを発生させます。`RabbitMQDiskFreeLimit` は Amazon MQ によって設定され、各ブローカーインスタンスタイプで使用可能なディスク容量を考慮して定義されています。

ディスク制限アラームを生成した Amazon MQ 上の RabbitMQ ブローカーは、新しいメッセージを発行できなくなります。同じ接続上にパブリッシャーとコンシューマーがある場合、コンシューマーもメッセージを受信できなくなります。RabbitMQ をクラスターで実行する場合、ディスクアラームはクラスター全体に適用されます。1 つのノードが制限を下回ると、他のすべてのノードが受信メッセージをブロックします。ディスク容量の不足のために、ブローカーではアラームの診断および解決を困難にする他の問題が発生することがあります。

Amazon MQ では、ディスクアラームが発生しているブローカーの再起動は行われません。また、ブローカーでアラームが発生し続ける限り `RebootBroker` API オペレーションに対して例外が返されます。

> **Note**
>
> ブローカーを mq.m5 インスタンスタイプから mq.t3.micro インスタンスタイプにダウングレードすることはできません。ダウングレードするには、ブローカーを削除し、新しいブローカーを作成する必要があります。

ディスク制限アラームの診断と対処

Amazon MQ は、デフォルトでブローカーのメトリクスを有効にします。Amazon CloudWatch コンソールにアクセスするか、CloudWatch API を使用して、ブローカーのメトリクスを表示できます。MessageCount は、RabbitMQ ディスク制限アラームを診断する際に役立つメトリクスです。メッセージは、消費または破棄されるまでメモリに格納されます。メッセージ数が多い場合は、ディスクストレージが過剰に使用されていることを示し、ディスクアラームの原因となる可能性があります。

ディスク制限アラームを診断するには、Amazon MQ マネジメントコンソールを使用して次の操作を行います。

- キューに発行されたメッセージを消費する新しい接続を作成します。
- キューからメッセージをパージします。
- ブローカーからキューを削除します。

> **Note**
>
> 必要なアクションを実行した後、RABBITMQ_DISK_ALARM ステータスがクリアされるまでに数時間かかる場合があります。

ディスク制限アラームの再発を防ぐには、ホストインスタンスタイプを追加のリソースを含むインスタンスにアップグレードします。ブローカーのインスタンスタイプを更新する方法については、Amazon MQ REST API リファレンスの「UpdateBrokerInput」を参照してください。また、パブリッシャーとコンシューマーは別々の接続上に保持することをお勧めします。

Amazon MQ for RabbitMQ: インスタンスタイプ変更アラーム

RABBITMQ_CLUSTER_DISK_USAGE_TOO_HIGH_FOR_INSTANCE_CHANGE は、現在の RabbitMQ ノードでディスク使用率が高いため、リクエストされたブローカーインスタンスタイプの変更を続行できないことを示します。Amazon MQ for RabbitMQ は、現在のディスク使用量が CloudWatch メトリクス RabbitMQDiskFree で識別されるリクエストされたインスタンスタイプで使用可能な容量を超えると、このアラームを生成します。

RABBITMQ_CLUSTER_DISK_USAGE_TOO_HIGH_FOR_INSTANCE_CHANGE 状態になった RabbitMQ ブローカーは引き続きアプリケーションで使用できますが、リクエストされたインスタンスタイプの変更は続行されません。Amazon MQ はこの状態でブローカーの再起動を許可しますが、ディスク使用量がリクエストされたインスタンスタイプのしきい値を上回っている間はインスタンスタイプを変更することはできません。ブローカーは、この状態のときにインスタンスタイプを変更しようとする ModifyBroker API オペレーションの例外を返します。

インスタンスタイプ変更アラームの診断と対処

Amazon MQ は、デフォルトでブローカーのメトリクスを有効にします。CloudWatch コンソールにアクセスするか、CloudWatch API を使用して、ブローカーのメトリクスを表示できます。MessageCount と RabbitMQDiskFree メトリクスを使用して RABBITMQ_CLUSTER_DISK_USAGE_TOO_HIGH_FOR_INSTANCE_CHANGE を診断できます。

隔離状態を解決し、インスタンスタイプの変更を続行できるようにするには、Amazon MQ マネジメントコンソールを使用して以下を行います。

- キューに発行されたメッセージを消費する新しい接続を作成します。
- キューからメッセージをパージします。
- ブローカーからキューを削除します。

> **Note**
>
> 必要なアクションを実行した後、RABBITMQ_CLUSTER_DISK_USAGE_TOO_HIGH_FOR_INSTANCE_CHANGE ステータスがクリアされるまでに数時間かかる場合があります。

Amazon MQ の RabbitMQ: 無効な IAM 継承ロール Amazon MQ

Amazon MQ の RabbitMQ は、 で指定された IAM ロール ARN が無効`aws.arns.assume_role_arn`であるか、Amazon MQ が引き受けることができない場合に、INVALID_ASSUMEROLE の重要なアクション必須コードを生成します。これは、ロールが存在しない場合、ブローカーとは異なる AWS アカウントにある場合、または mq.amazonaws.com と必要な信頼関係がない場合に発生する可能性があります。

RABBITMQ_INVALID_ASSUMEROLE 隔離のブローカーは、LDAP 認証に必要な認証情報または証明書を取得できず、LDAP 認証が使用できなくなります。LDAP が唯一の設定済み認証方法である場合、ユーザーはブローカーに接続できません。IAM ロールは、Amazon MQ が、LDAP 認証に使用される AWS Secrets Manager シークレットや Amazon S3 オブジェクトなど、ブローカー設定の ARNs によって参照される AWS リソースにアクセスするために必要です。

RABBITMQ_INVALID_ASSUMEROLE の診断と対処

RABBITMQ_INVALID_ASSUMEROLE アクションの必須コードを診断して対処するには、Amazon CloudWatch Logs と AWS Identity and Access Management コンソールを使用する必要があります。

無効な継承ロールの問題を解決するには

1. Amazon CloudWatch Logs Insights に移動し、ブローカーのロググループ に対して次のクエリを実行します`/aws/amazonmq/broker/<broker-id>/general`。

```
fields @timestamp, @message
| sort @timestamp desc
| filter @message like /error.*aws_arn_config/
| limit 10000
```

2. 次のようなエラーメッセージを探します。

```
[error] <0.254.0> aws_arn_config: {handle_assume_role,{error,
{assume_role_failed,"AWS service is unavailable"}}}
```

3. IAM ロールの設定を確認し、次のような問題を修正します。

- ロールがブローカーと同じ AWS アカウントに存在することを確認する
- 信頼ポリシーが mq.amazonaws.com にロールの引き受けを許可することを確認する
- ロールに必要な AWS リソースにアクセスするための適切なアクセス許可があることを確認します。

4. ブローカー設定を更新する前に、ARN アクセス検証 API エンドポイントを使用して修正を検証します。
5. ブローカー設定を更新し、ブローカーを再起動します。

Amazon MQ の RabbitMQ: 無効な LDAP ARN Amazon MQ

Amazon MQ の RabbitMQ は、LDAP サービスアカウントのパスワード用に設定された ARN が無効またはアクセスできない場合、INVALID_ARN_LDAP の重要なアクション必須コードを生成します。これは、プレーンテキストパスワードを含む AWS Secrets Manager シークレットを参照する必要がある `aws.arns.auth_ldap.other_bind.passwordaws.arns.auth_ldap.dn_lookup_bind.password` たは で指定された ARNs に適用されます。

RABBITMQ_INVALID_ARN_LDAP 隔離のブローカーは LDAP サービスアカウントで認証できないため、LDAP 認証を使用できません。LDAP が唯一の設定済み認証方法である場合、ユーザーはブローカーに接続できません。無効な ARNs は、不正な形式の ARN 構文、存在しないシークレットへの参照、ブローカーとは異なる AWS リージョンにあるシークレット、または IAM ロールの Secretsmanager:GetSecretValue アクセス許可が不十分であることが原因で発生する可能性があります。

RABBITMQ_INVALID_ARN_LDAP の診断と対処

RABBITMQ_INVALID_ARN_LDAP アクションの必須コードを診断して対処するには、Amazon CloudWatch Logs と コンソールを使用する必要があります。

無効な LDAP ARN の問題を解決するには

1. Amazon CloudWatch Logs Insights に移動し、ブローカーのロググループ に対して次のクエリを実行します`/aws/amazonmq/broker/<broker-id>/general`。

```
fields @timestamp, @message
| sort @timestamp desc
```

```
| filter @message like /error.*aws_arn_config/
| limit 10000
```

2. 次のようなエラーメッセージを探します。

```
[error] <0.254.0> aws_arn_config: {<<"could not resolve
 ARN 'arn:aws:secretsmanager:xxx' for configuration
 'aws.arns.auth_ldap.dn_lookup_bind.password', error: \"AWS service is unavailable
\"">>,{error,"AWS service is unavailable"}}
```

3. Secrets Manager シークレットを確認し、次のような問題を修正します。
   - シークレットがブローカーと同じ AWS リージョンに存在することを確認する
   - ARN 構文が正しいことを確認する
   - IAM ロールに secretsmanager:GetSecretValue アクセス許可があることを確認する
4. ブローカー設定を更新する前に、ARN アクセス検証 API エンドポイントを使用して修正を検証します。
5. ブローカー設定を更新し、ブローカーを再起動します。

Amazon MQ の RabbitMQ: 無効な HTTP ARN Amazon MQ

Amazon MQ の RabbitMQ は、HTTP auth_backend の SSL 証明書または
はキーファイルの 1 つ以上の ARNs が無効またはアクセスできない場
合、INVALID_ARN_HTTP の重要なアクションに必要なコードを生成します。これ
は`aws.arns.auth_http.ssl_options.cacertfile`、、`aws.arns.auth_http.ssl_options.cer`
たは で指定された ARNS に適用されます。ARNS
は`aws.arns.auth_http.ssl_options.keyfile`、証明書とプライベートキーを含む Amazon
S3 オブジェクトと AWS Secrets Manager シークレットを参照する必要があります。

RABBITMQ_INVALID_ARN_HTTP 隔離のブローカーは、HTTP サーバー経由で認証できません。HTTP が唯一の設定済み認証方法である場合、ユーザーはブローカーに接続できません。無効な ARNs は、不正な形式の ARN 構文、存在しないシークレットへの参照、ブローカーとは異なる AWS リージョンにあるシークレット、または IAM ロールの s3:GetObject/secretsmanager:GetSecretValue アクセス許可が不十分であることが原因で発生する可能性があります。

RABBITMQ_INVALID_ARN_HTTP の診断と対処

RABBITMQ_INVALID_ARN_HTTP アクションの必須コードを診断して対処するには、Amazon CloudWatch Logs と コンソールを使用する必要があります。

無効な HTTP ARN の問題を解決するには

1. Amazon CloudWatch Logs Insights に移動し、ブローカーのロググループ に対して次のクエリを実行します`/aws/amazonmq/broker/<broker-id>/general`。

```
fields @timestamp, @message
| sort @timestamp desc
| filter @message like /error.*aws_arn_config/
| limit 10000
```

2. 次のようなエラーメッセージを探します。

```
[error] <0.209.0> aws_arn_config: {<<"could not resolve ARN 'arn:aws:s3:::xxxx' for
 configuration 'aws.arns.auth_http.ssl_options.certfile', error: \"AWS service is
 unavailable\"">>,{error,"AWS service is unavailable"}}
```

3. S3 Object/Secrets Manager シークレットをチェックし、次のような問題を修正します。
   - リソースがブローカーと同じ AWS リージョンに存在することを確認する
   - ARN 構文が正しいことを確認する
   - IAM ロールに s3:GetObject および secretsmanager:GetSecretValue アクセス許可があることを確認します。
4. ブローカー設定を更新する前に、ARN アクセス検証 API エンドポイントを使用して修正を検証します。
5. ブローカー設定を更新し、ブローカーを再起動します。

Amazon MQ の RabbitMQ: 無効な SSL ARN Amazon MQ

Amazon MQ の RabbitMQ は、EXTERNAL auth_mechanism の CA 証明書トラストストアの 1 つ以上の ARNs が無効またはアクセスできない場合、INVALID_ARN_SSL の重要なアクションに必

要なコードを生成します。これは、 aws.arns.ssl_options.cacertfileまたは で指定された ARNS に適用されます。ARNS はaws.arns.management.ssl.cacertfile、証明書を含む Amazon S3 または ACM PCA オブジェクトを参照する必要があります。

RABBITMQ_INVALID_ARN_SSL 隔離のブローカーは、有効なトラストストアが設定されていないため、相互 TLS ハンドシェイク中にクライアント証明書を認証できません。EXTERNAL 認証メカニズムが唯一の設定済み認証方法である場合、ユーザーはブローカーに接続できません。無効な ARNs は、不正な形式の ARN 構文、存在しない S3 オブジェクトへの参照、ブローカーとは異なる AWS リージョンにある S3 オブジェクト、または IAM ロールの s3:GetObject/acm-pca:GetCertificateAuthorityCertificate アクセス許可が不十分であることが原因で発生する可能性があります。

RABBITMQ_INVALID_ARN_SSL の診断と対処

RABBITMQ_INVALID_ARN_SSL アクションの必須コードを診断して対処するには、Amazon CloudWatch Logs と コンソールを使用する必要があります。

無効な SSL ARN の問題を解決するには

1. Amazon CloudWatch Logs Insights に移動し、ブローカーのロググループ に対して次のクエリを実行します/aws/amazonmq/broker/<broker-id>/general。

```
fields @timestamp, @message
| sort @timestamp desc
| filter @message like /error.*aws_arn_config/
| limit 10000
```

2. 次のような エラーメッセージを探します。

```
[error] <0.209.0> aws_arn_config: {<<"could not resolve ARN 'arn:aws:acm-pca:xxxx'
 for configuration 'aws.arns.ssl_options.cacertfile', error: \"AWS service is
 unavailable\"">>,{error,"AWS service is unavailable"}}
```

3. S3/ACM-PCA オブジェクトを確認し、次のような問題を修正します。

   - シークレットがブローカーと同じ AWS リージョンに存在することを確認する
   - ARN 構文が正しいことを確認する

- IAM ロールに s3:GetObject/acm-pca:GetCertificateAuthorityCertificate アクセス許可があることを確認します。

4. ブローカー設定を更新する前に、ARN アクセス検証 API エンドポイントを使用して修正を検証します。
5. ブローカー設定を更新し、ブローカーを再起動します。

Amazon MQ の RabbitMQ: 無効な ARN Amazon MQ

Amazon MQ の RabbitMQ は、ブローカーで設定された 1 つ以上の ARNs が無効またはアクセスできない場合、INVALID_ARN の重要なアクション必須コードを生成します。これは、SSL 証明書、 AWS Secrets Manager シークレット、Amazon S3 オブジェクト、または RABBITMQ_INVALID_ARN_LDAP や RABBITMQ_INVALID_ASSUMEROLE など、より具体的な隔離コードの対象ではないその他の AWS リソース参照に使用される ARNs に適用されます。

RABBITMQ_INVALID_ARN 隔離のブローカーではARNs が無効であるかに応じて、機能が低下する可能性があります。アクセスできないリソースに依存する機能は使用できなくなり、ブローカーは解決に失敗した ARN を示すエラーをログに記録します。ブローカーの可用性への影響は、重要なブローカーオペレーションに無効な ARN が必要かどうかによって異なります。

RABBITMQ_INVALID_ARN の診断と対処

RABBITMQ_INVALID_ARN アクションの必須コードを診断して対処するには、Amazon CloudWatch Logs と、影響を受けるリソースの適切な AWS サービスコンソールを使用する必要があります。

無効な ARN の問題を解決するには

1. Amazon CloudWatch Logs Insights に移動し、ブローカーのロググループ に対して次のクエリを実行します`/aws/amazonmq/broker/<broker-id>/general`。

```
fields @timestamp, @message
| sort @timestamp desc
| filter @message like /error.*aws_arn_config/
| limit 10000
```

2. 次のようなエラーメッセージを探します。

```
[error] <0.254.0> aws_arn_config: {<<"could not resolve ARN
 'arn:aws:s3:::bucket-name/certificate.pem' for configuration
 'aws.arns.auth_ldap.ssl_options.cacertfile', error: \"AWS service is unavailable
\"">>,{error,"AWS service is unavailable"}}
```

3. AWS リソースを確認し、次のような問題を修正します。
   - リソースがブローカーと同じ AWS リージョンに存在することを確認する
   - ARN 構文が正しいことを確認する
   - IAM ロールにリソースにアクセスするための適切なアクセス許可があることを確認します。
4. ブローカー設定を更新する前に、ARN アクセス検証 API エンドポイントを使用して修正を検証します。
5. ブローカー設定を更新し、ブローカーを再起動します。

関連リソース

Amazon MQ のリソース

以下の表は、Amazon MQ の使用に役立つリソースのリストです。

リソース	説明
Amazon MQ REST API リファレンス	REST リソース、サンプルリクエスト、HTTP メソッド、スキーマ、パラメータ、およびサービスから返されるエラーの説明です。
「AWS CLI コマンドリファレンス」内の Amazon MQ	メッセージブローカーで使用できる AWS CLI コマンドの説明です。
「AWS CloudFormation ユーザーガイド」内の Amazon MQ	`AWS::Amazon MQ::Broker` リソースを使用すると、Amazon MQ ブローカーを作成する、指定されたブローカーに対して設定変更の追加またはユーザーの変更を行う、指定されたブローカーに関する情報を返す、および指定されたブローカーを削除することができます。 `AWS::Amazon MQ::Configuration` リソースを使用すると、Amazon MQ 設定を作成する、設定変更の追加とユーザーの変更を行う、および指定された設定に関する情報を返すことができます。
のリージョンとエンドポイント	Amazon MQ のリージョンとエンドポイントに関する情報
製品ページ	Amazon MQ に関する情報のメインウェブページです。
ディスカッションフォーラム	デベロッパーが Amazon MQ に関連する技術的な質問について話し合うためのコミュニティベースのフォーラムです。

リソース	説明
AWS プレミアムサポート情報	AWS のインフラストラクチャサービスでのアプリケーションの構築と実行を支援するための、1 対 1 で対応が迅速なサポートチャネル、AWS Premium Support サポートに関する情報のメインウェブページです。

Amazon MQ for ActiveMQ のリソース

以下の表は、Apache ActiveMQ の使用に役立つリソースのリストです。

リソース	説明
Apache ActiveMQ Getting Started Guide	Apache ActiveMQ の公式ドキュメントです。
ActiveMQ in Action	JMS メッセージ、コネクタ、メッセージの持続性、認証、認可の構造を説明した Apache ActiveMQ のガイドです。
言語間のクライアント	プログラミング言語と対応する Apache ActiveMQ ライブラリのリストです。「ActiveMQ クライアント」と「QpidJMS クライアント」も参照してください。

Amazon MQ for RabbitMQ のリソース

以下の表は、RabbitMQ の使用に役立つリソースのリストです。

リソース	説明
The RabbitMQ Getting Started Guide	RabbitMQ の公式ドキュメントです。
RabbitMQ Client Libraries and Developer Tools	さまざまなプログラミング言語とプラットフォームを使用した RabbitMQ での作業のための、公式にサポートされているクライアントラ

リソース	説明
	イブラリとデベロッパーツールに関するガイドです。
RabbitMQ Best Practices	RabbitMQ を使用するためのベストプラクティスと推奨事項に関する CloudAMQP のガイドです。

Amazon MQ リリースノート

以下の表には、Amazon MQ 機能のリリースおよび改善がリストされています。

日付	ドキュメントの更新
2026 年 2 月 19 日	Amazon MQ Amazon MQ は、新しいマイナーエンジンバージョンのリリースである ActiveMQ 5.19 をサポートするようになりました。 詳細については、以下を参照してください。 • ActiveMQ 5.19 リリースページ • Amazon MQ for ActiveMQ エンジンバージョンの管理 • Amazon MQ ブローカーエンジンバージョンのアップグレード • Spring XML 設定ファイルの使用
2026 年 1 月 22 日	Amazon MQ は、RabbitMQ 4.2 以降のブローカーの JMS トピック交換プラグインをサポートするようになりました。公式 RabbitMQ JMS クライアントを使用して、Amazon MQ for RabbitMQ ブローカーで JMS ワークロードを実行できます。JMS 1.1、2.0、3.1 をサポートしています。 詳細については、以下を参照してください。 • 公式 JMS 2.0 仕様 (および拡張 JMS 1.1 との下位互換性) • 公式 JMS 3.1 仕様 • RabbitMQ JMS クライアントの制限 • JMS アプリケーションを Amazon MQ for RabbitMQ ブローカーに接続する
2026 年 1 月 8 日	Amazon MQ は、X.509 クライアント証明書を使用した RabbitMQ 4.2 以降のブローカーの SSL 証明書認証と相互 TLS (mTLS) 設定をサポートするようになりました。SSL 証明書認証と mTLS は AWS マネジメントコンソール、、AWS CloudFormation AWS CLI、または Amazon MQ が利用可能なすべての AWS リージョン AWS CDK で設定できます。 詳細については、「SSL 証明書認証」および「mTLS の設定」を参照してください。

日付	ドキュメントの更新
2026 年 1 月 6 日	Amazon MQ は、外部 HTTP サーバーを使用する RabbitMQ 4.2 以降のブローカーの HTTP 認証と認可をサポートするようになりました。HTTP 認証は AWS マネジメントコンソール、、 AWS CloudFormation AWS CLI、または Amazon MQ が利用可能なすべての AWS リージョン AWS CDK で設定できます。 詳細については、「HTTP 認証と認可」を参照してください。
2025 年 11 月 20 日	Amazon MQ は、AMQP 1.0 プロトコルのネイティブサポート、新しい Raft ベースのメタデータストア Khepri、ローカルシャベル、クォーラムキューのメッセージ優先順位を導入した新しいメジャーバージョンリリースである RabbitMQ 4.2 をサポートするようになりました。RabbitMQ 4.2 には、スループットとメモリ管理のためのさまざまなバグ修正とパフォーマンスの向上も含まれています。このバージョンでは新機能が導入されていますが、いくつかの重大な変更があります。 詳細については、以下を参照してください。 • RabbitMQ 4 • オープンソース RabbitMQ リリースノート • リソース制限の設定 • サポートされているプロトコル • Amazon MQ バージョンのアップグレード
2024 年 11 月 18 日	Amazon MQ は、アフリカ (ケープタウン) で、中規模から 16xlarge までのサイズの RabbitMQ 用の Graviton3 搭載の m7g インスタンスをサポートするようになりました。 詳細については、「Amazon MQ for RabbitMQ ブローカーのインスタンスタイプ」を参照してください。

日付	ドキュメントの更新
2025 年 11 月 17 日	Amazon MQ は、外部 LDAP ディレクトリサービスを持つ RabbitMQ ブローカーの LDAP 認証と認可をサポートするようになりました。LDAP は AWS マネジメントコンソール、、 AWS CloudFormation AWS CLI、または Amazon MQ が利用可能なすべての AWS リージョン AWS CDK で設定できます。 詳細については、「Amazon MQ for RabbitMQ の LDAP 認証と認可」を参照してください。
2025 年 10 月 22 日	Amazon MQ がアジアパシフィック (ニュージーランド) リージョンで使用可能になりました。 利用可能なリージョンの詳細については、AWS 全般リファレンスガイドの「AWS リージョンとエンドポイント」を参照してください。
2025 年 9 月 3 日	Amazon MQ は、パブリック ID プロバイダー (IdP) を使用する RabbitMQ ブローカーの OAuth 2.0 認証と認可をサポートするようになりました。OAuth 2.0 は AWS マネジメントコンソール、、 AWS CloudFormation AWS CLI、または Amazon MQ が利用可能なすべての AWS リージョン AWS CDK で設定できます。 詳細については、「Amazon MQ for RabbitMQ に対する OAuth 2.0 の認証と認可」を参照してください。
2025 年 7 月 22 日	Amazon MQ は、中規模から 16xlarge のサイズ範囲で RabbitMQ の Graviton3 搭載 `m7g` インスタンスをサポートするようになりました。`m7g` インスタンスで実行されている RabbitMQ クラスターは、`m5` インスタンスで実行されている同等の Amazon MQ for RabbitMQ クラスターと比較して、ワークロード容量が最大 50% 増加し、スループットが最大 85% 向上します。 `M7g` インスタンスには、インスタンスサイズによって異なる最適化されたディスクボリュームサイズもあります。詳細については、「Broker instance types」を参照してください。 Amazon MQ の `M7g` インスタンスは、現在、アフリカ (ケープタウン)、カナダ西部 (カルガリー)、欧州 (ミラノ) の各リージョンを除く、一般的に利用可能なすべてのリージョンで利用できます。

日付	ドキュメントの更新
2025 年 7 月 8 日	Amazon MQ がアジアパシフィック (台北) リージョンで利用可能になりました。 利用可能なリージョンの詳細については、AWS 全般リファレンスガイドの「AWS リージョンとエンドポイント」を参照してください。
2025 年 4 月 22 日	`DeleteConfiguration` API を使用して Amazon MQ ブローカー設定を削除できるようになりました。詳細については、「Amazon MQ API リファレンス」の「設定」を参照してください。
2025 年 4 月 16 日	Amazon MQ for RabbitMQ で、デュアルスタック (IPv4 および IPv6) エンドポイントを使用してパブリックブローカーとプライベートブローカーに接続できるようになりました。詳細については、「Connecting to Amazon MQ」および「Configuring a private Amazon MQ broker」を参照してください。
2025 年 4 月 7 日	Amazon MQ が、アジアパシフィック (タイ) およびメキシコ (中部) リージョンで利用可能になりました。 利用可能なリージョンの詳細については、AWS 全般リファレンスガイドの「AWS リージョンとエンドポイント」を参照してください。
2025 年 2 月 13 日	Amazon MQ API FIPS エンドポイントが、カナダ (中部) およびカナダ西部 (カルガリー) リージョンで利用可能になりました。 Amazon MQ API で FIPS エンドポイントを使用する方法の詳細については、「Connecting to Amazon MQ」を参照してください。 利用可能なリージョンの詳細については、AWS 全般リファレンスガイドの「AWS リージョンとエンドポイント」を参照してください。

日付	ドキュメントの更新
2025 年 2 月 12 日	Amazon MQ は、次のインスタンスタイプのサポート終了日を発表しました。 Broker instance types • ActiveMQ `mq.t2.micro` : 2025 年 5 月 12 日 • ActiveMQ `mq.m4.large` : 2025 年 5 月 12 日 2025 年 3 月 17 日以降は、`mq.t2.micro` または `mq.m4.large` でブローカーを作成できません。
2024 年 12 月 10 日	Amazon MQ は AWS PrivateLink を使用して、トラフィックをパブリックインターネットに公開することなく、仮想プライベートクラウド (VPCs) と Amazon MQ API 間の接続をサポートするようになりました。詳細については、「the section called “AWS PrivateLink を使用して Amazon MQ に接続する”」を参照してください。
2024 年 11 月 18 日	Amazon MQ がアジアパシフィック (マレーシア) リージョンで利用可能になりました。利用可能なリージョンの詳細については、AWS 全般リファレンスガイドの「AWS リージョンとエンドポイント」を参照してください。
2024 年 11 月 14 日	Amazon MQ は、次のエンジンバージョンのサポート終了日を発表しました。 Amazon MQ for ActiveMQ エンジンバージョンの管理 • ActiveMQ 5.17: 2025 年 6 月 16 日 Amazon MQ for RabbitMQ エンジンバージョンの管理 • RabbitMQ 3.11: 2025 年 2 月 17 日 • RabbitMQ 3.12: 2025 年 3 月 17 日 最新バージョンにアップグレードする方法の詳細については、「Amazon MQ ブローカーエンジンバージョンのアップグレード」を参照してください。

日付	ドキュメントの更新
2024 年 11 月 13 日	Amazon MQ は、IPv4 または IPv6 を使用して接続できるデュアルスタックサービスエンドポイントをサポートするようになりました。Amazon MQ デュアルスタックリージョンサービスエンドポイントは、A と AAAA 両方の DNS レコードで解決できます。詳細については、「???」を参照してください。
2024 年 7 月 25 日	Amazon MQ が、新しいマイナーエンジンバージョンのリリースである ActiveMQ 5.18 をサポートするようになりました。詳細については次を参照してください: • ActiveMQ 5.18 リリースページ • Amazon MQ for ActiveMQ エンジンバージョンの管理 • Amazon MQ ブローカーエンジンバージョンのアップグレード • Spring XML 設定ファイルの使用
2024 年 7 月 22 日	Amazon MQ が、バージョン 3.13 以降のブローカーでのみクォーラムキューをサポートするようになりました。クォーラムキューは、Raft コンセンサスアルゴリズムを使用してデータ整合性を維持する、レプリケーション型の FIFO キュータイプです。クォーラムキューは有害メッセージの処理を提供するため、未処理のメッセージの管理に役立つ可能性があります。 クォーラムキューの使用を開始するには、「Amazon MQ での RabbitMQ のクォーラムキュー」を参照してください。

日付	ドキュメントの更新
2024 年 7 月 2 日	Amazon MQ for RabbitMQ が、マイナーバージョンリリースである RabbitMQ 3.13 をサポートするようになりました。エンジンバージョン 3.13 以降を使用しているすべてのブローカーについて、Amazon MQ は、サポートされている最新のパッチバージョンへのアップグレードをメンテナンスウィンドウ内で管理します。詳細については、「Amazon MQ ブローカーエンジンバージョンのアップグレード」を参照してください。 「Amazon MQ for RabbitMQ のサイズ設定ガイドライン」が更新され、エンジンバージョン 3.13 を使用するブローカーのキュー、チャネルあたりのコンシューマー、シャベルの新しい制限値が含められました。 このリリースでの修正と機能の詳細については、RabbitMQ サーバー GitHub リポジトリの RabbitMQ 3.13 リリースノートを参照してください。 サポートされる Amazon MQ for RabbitMQ のバージョンとブローカーアップグレードの詳細については、「Amazon MQ for RabbitMQ エンジンバージョンの管理」を参照してください。
2024 年 6 月 10 日	Amazon MQ がカナダ西部 (カルガリー) リージョンで利用可能になりました。利用可能なリージョンの詳細については、AWS 全般リファレンスガイドの「AWS リージョンとエンドポイント」を参照してください。

日付	ドキュメントの更新
2024 年 5 月 10 日	Amazon MQ バージョンサポートカレンダーは、ブローカーエンジンバージョンがサポート終了に達するタイミングを示します。あるエンジンバージョンがサポート終了に達すると、Amazon MQ は、そのバージョンのすべてのブローカーを、サポートされている次のマイナーバージョンに自動的に更新します。Amazon MQ は、エンジンバージョンがサポート終了に達する少なくとも 90 日前に通知を送信します。 バージョンサポートカレンダーとサポート終了日を確認するには、以下を参照してください。 • Amazon MQ for ActiveMQ エンジンバージョンの管理 • Amazon MQ for RabbitMQ エンジンバージョンの管理 自動マイナーバージョンアップグレードを有効にして、メンテナンスウィンドウ内でブローカーが次のパッチバージョンに更新されるようにすることもできます。詳細については、Amazon MQ ブローカーエンジンバージョンのアップグレードを参照してください。
2024 年 5 月 9 日	Amazon MQ for RabbitMQ が、マイナーバージョンリリースである RabbitMQ 3.12 をサポートするようになりました。3.12.13 以降のすべてのブローカーは Classic Queues バージョン 2 (CQv2) を使用し、3.12.13 以降のすべてのキューはレイジーキューとして動作します。 3.12.13 より前のバージョンのブローカーは、CQv2 キューとレイジーキューを有効にするか、Amazon MQ for RabbitMQ の最新バージョンにアップグレードすることをお勧めします。 このリリースでの修正と機能の詳細については、以下を参照してください。 • RabbitMQ サーバー GitHub リポジトリの RabbitMQ 3.12 リリースノート サポートされる Amazon MQ for RabbitMQ のバージョンとブローカーアップグレードの詳細については、「Amazon MQ for RabbitMQ エンジンバージョンの管理」を参照してください。

日付	ドキュメントの更新
2024 年 3 月 4 日	Amazon MQ for RabbitMQ が RabbitMQ 3.11.28 をサポートするようになりました。 このリリースでの修正と機能の詳細については、以下を参照してください。 • RabbitMQ サーバー GitHub リポジトリの RabbitMQ 3.11.28 リリースノート • RabbitMQ changelog サポートされる Amazon MQ for RabbitMQ のバージョンとブローカーアップグレードの詳細については、「Amazon MQ for RabbitMQ エンジンバージョンの管理」を参照してください。
2024 年 1 月 19 日	Amazon MQ for RabbitMQ では、ユーザー名「guest」はサポートされず、デフォルトのゲストアカウントは新しいブローカーの作成時に削除されます。ユーザーが作成した「guest」というアカウントも、Amazon MQ によって定期的に削除されます。
2023 年 12 月 15 日	Amazon MQ がイスラエル (テルアビブ) リージョンで利用可能になりました。利用可能なリージョンの詳細については、AWS 全般リファレンスガイドの「AWS リージョンとエンドポイント」を参照してください。
2023 年 12 月 11 日	Amazon MQ for RabbitMQ が RabbitMQ 3.10.25 をサポートするようになりました。 このリリースでの修正と機能の詳細については、以下を参照してください。 • RabbitMQ サーバー GitHub リポジトリの RabbitMQ 3.10.25 リリースノート • RabbitMQ changelog サポートされる Amazon MQ for RabbitMQ のバージョンとブローカーアップグレードの詳細については、「Amazon MQ for RabbitMQ エンジンバージョンの管理」を参照してください。

日付	ドキュメントの更新
2023 年 10 月 26 日	Amazon MQ は、重要な更新を含む最新の ActiveMQ マイナーバージョン 5.15.16、5.16.7、5.17.6 をリリースしました。ActiveMQ の古いマイナーバージョンを非推奨とし、すべてのブローカーについて 5.15 のすべてのバージョンを 5.15.16、5.16 のすべてのバージョンを 5.16.7、5.17 のすべてのバージョンを 5.17.6 にアップデートします。 ActiveMQ ブローカーの更新の詳細については、「Amazon MQ for ActiveMQ エンジンバージョンの管理」を参照してください。
2023 年 9 月 27 日	Amazon MQ for RabbitMQ が RabbitMQ 3.11.20 をサポートするようになりました。 このリリースでの修正と機能の詳細については、以下を参照してください。 • RabbitMQ サーバー GitHub リポジトリの「RabbitMQ 3.11.20 リリースノート」 • RabbitMQ changelog サポートされる Amazon MQ for RabbitMQ のバージョンとブローカーアップグレードの詳細については、「Amazon MQ for RabbitMQ エンジンバージョンの管理」を参照してください。
2023 年 7 月 27 日	Amazon MQ for RabbitMQ が RabbitMQ 3.11.16 をサポートするようになりました。 このリリースでの修正と機能の詳細については、以下を参照してください。 • RabbitMQ サーバー GitHub リポジトリの RabbitMQ 3.11.16 リリースノート • RabbitMQ changelog サポートされる Amazon MQ for RabbitMQ のバージョンとブローカーアップグレードの詳細については、「Amazon MQ for RabbitMQ エンジンバージョンの管理」を参照してください。

日付	ドキュメントの更新
2023 年 7 月 27 日	Amazon MQ for RabbitMQ は、RabbitMQ ブローカーの設定の作成と適用をサポートするようになりました。 ブローカーに設定を追加する方法の詳細については、「RabbitMQ Broker Configurations」を参照してください。 この機能の詳細については、以下を参照してください。 • オペレーターポリシー • オペレーターポリシーの変更
2023 年 6 月 23 日	Amazon MQ が、新しいマイナーエンジンバージョンのリリースである ActiveMQ 5.17.3 をサポートするようになりました。このリリースでは、Amazon MQ の新しいクロスリージョンデータレプリケーション (CRDR) 機能をサポートしています。 詳細については次を参照してください: • CRDR の開始方法については、開発者ガイドの「Amazon MQ for ActiveMQ のクロスリージョンデータレプリケーション」を参照してください。 • ActiveMQ 5.17.3 リリースページ • Amazon MQ for ActiveMQ エンジンバージョンの管理 • Amazon MQ ブローカーエンジンバージョンのアップグレード • Spring XML 設定ファイルの使用
2023 年 6 月 21 日	Amazon MQ for ActiveMQ は、クロスリージョンデータレプリケーション (CRDR) 機能を提供するようになりました。これにより、プライマリリージョンのプライマリブローカーからレプリカ AWS リージョンのレプリカブローカーへの非同期メッセージレプリケーションが可能になります。プライマリリージョンのプライマリブローカーに障害が発生した場合、スイッチオーバーまたはフェイルオーバーを開始することで、セカンダリリージョンのレプリカブローカーをプライマリに昇格させることができます。 CRDR の開始方法については、開発者ガイドの「Amazon MQ for ActiveMQ のクロスリージョンデータレプリケーション」を参照してください。

日付	ドキュメントの更新
2023 年 5 月 18 日	Amazon MQ は、以下のリージョンでご利用いただけるようになりました。 • アジアパシフィック (メルボルン) • アジアパシフィック (ハイデラバード) • 欧州 (スペイン) • 欧州 (チューリッヒ) 利用可能なリージョンの詳細については、AWS 全般リファレンスガイドの「AWS リージョンとエンドポイント」を参照してください。
2023 年 4 月 14 日	Amazon MQ for RabbitMQ が RabbitMQ バージョン 3.9.27 をサポートするようになりました。 このリリースでの修正と機能の詳細については、以下を参照してください。 • RabbitMQ サーバー GitHub リポジトリの RabbitMQ 3.9.27 リリースノート • RabbitMQ changelog サポートされる Amazon MQ for RabbitMQ のバージョンとブローカーアップグレードの詳細については、「Amazon MQ for RabbitMQ エンジンバージョンの管理」を参照してください。
2023 年 4 月 14 日	Amazon MQ for RabbitMQ が RabbitMQ バージョン 3.10.20 をサポートするようになりました。 このリリースでの修正と機能の詳細については、以下を参照してください。 • RabbitMQ サーバー GitHub リポジトリの RabbitMQ 3.10.20 リリースノート • RabbitMQ changelog サポートされる Amazon MQ for RabbitMQ のバージョンとブローカーアップグレードの詳細については、「Amazon MQ for RabbitMQ エンジンバージョンの管理」を参照してください。

日付	ドキュメントの更新
2023 年 3 月 31 日	Amazon MQ for RabbitMQ が RabbitMQ エンジンバージョン 3.10.17 を無効化 Amazon MQ for RabbitMQ チームと RabbitMQ のオープンソース保守管理者は、バージョン 3.10.17 の RabbitMQ マネジメントコンソールに関する問題を特定しました。Amazon MQ はこのバージョンを撤回しました。この問題の影響を軽減するために、RabbitMQ の新しいパッチバージョンのサポートに取り組んでいる間、バージョン 3.10.10 で新しいブローカーを作成してください。バージョンアップグレードオプションをアクティブ化して、最新のバグ修正、セキュリティ更新、パフォーマンス強化を自動的に取得することをお勧めします。 Amazon MQ for RabbitMQ の利用可能なバージョンの詳細については、「Amazon MQ for RabbitMQ エンジンバージョン」を参照してください。
2023 年 3 月 1 日	Amazon MQ for RabbitMQ が RabbitMQ バージョン 3.10.17 をサポートするようになりました。 このリリースでの修正と機能の詳細については、以下を参照してください。 • RabbitMQ サーバー GitHub リポジトリの RabbitMQ 3.10.17 リリースノート • RabbitMQ changelog サポートされる Amazon MQ for RabbitMQ のバージョンとブローカーアップグレードの詳細については、「Amazon MQ for RabbitMQ エンジンバージョンの管理」を参照してください。

日付	ドキュメントの更新
2023 年 2 月 21 日	Amazon MQ for RabbitMQ が AWS Key Management Service (KMS) と統合され、サーバー側の暗号化が提供されるようになりました。独自のカスタマーマネージド CMK を選択するか、 AWS KMS アカウントで AWS マネージド KMS キーを使用できるようになりました。詳細については、「保管中の暗号化」を参照してください。 Amazon MQ は、次の方法で AWS KMS キーの使用をサポートしています。 • Amazon MQ 所有の KMS キー (デフォルト) – キーは Amazon MQ が所有、管理し、ユーザーのアカウントにはありません。 • AWS マネージド KMS キー — AWS マネージド KMS キー (aws/mq) は、Amazon MQ によってユーザーに代わって作成、管理、使用されるアカウントの KMS キーです。 • 既存のカスタマーマネージド KMS キーを選択する – カスタマーマネージド KMS キーは、ユーザーが AWS Key Management Service (KMS) で作成し、管理します。
2023 年 1 月 13 日	Amazon MQ for RabbitMQ が RabbitMQ バージョン 3.8.34 をサポートするようになりました。 このリリースでの修正と機能の詳細については、以下を参照してください。 • RabbitMQ サーバー GitHub リポジトリの RabbitMQ 3.8.34 リリースノート • RabbitMQ changelog サポートされる Amazon MQ for RabbitMQ のバージョンとブローカーアップグレードの詳細については、「Amazon MQ for RabbitMQ エンジンバージョンの管理」を参照してください。

日付	ドキュメントの更新
2022 年 12 月 15 日	Amazon MQ for RabbitMQ が RabbitMQ バージョン 3.9.24 をサポートするようになりました。 このリリースでの修正と機能の詳細については、以下を参照してください。 • RabbitMQ サーバー GitHub リポジトリの RabbitMQ 3.9.24 リリースノート • RabbitMQ changelog サポートされる Amazon MQ for RabbitMQ のバージョンとブローカーアップグレードの詳細については、「Amazon MQ for RabbitMQ エンジンバージョンの管理」を参照してください。
2022 年 12 月 13 日	Amazon MQ が、中東 (UAE) リージョンで利用可能になりました。利用可能なリージョンの詳細については、AWS 全般リファレンスガイドの「AWS リージョンとエンドポイント」を参照してください。
2022 年 11 月 14 日	Amazon MQ for RabbitMQ が、メジャーエンジンバージョンのリリースである 3.10 をサポートするようになりました。RabbitMQ キューで Queues バージョン 2 (CQv2) を有効にできるようになりました。3.8 から 3.10 への直接更新はサポートされていません。詳細については次を参照してください: • RabbitMQ 3.10.10 リリースノート • RabbitMQ changelog サポートされる Amazon MQ for RabbitMQ のバージョンとブローカーアップグレードの詳細については、「Amazon MQ for RabbitMQ エンジンバージョンの管理」を参照してください。

日付	ドキュメントの更新
2022 年 11 月 9 日	Amazon MQ が、新しいマイナーエンジンバージョンのリリースである ActiveMQ 5.17.2 をサポートするようになりました。詳細については次を参照してください: • ActiveMQ 5.17.2 リリースページ • Amazon MQ for ActiveMQ エンジンバージョンの管理 • Amazon MQ ブローカーエンジンバージョンのアップグレード • Spring XML 設定ファイルの使用
2022 年 8 月 17 日	Amazon MQ が、新しいメジャーエンジンバージョンのリリースである ActiveMQ 5.17.1 をサポートするようになりました。詳細については次を参照してください: • ActiveMQ 5.17.1 リリースページ • Amazon MQ for ActiveMQ エンジンバージョンの管理 • Amazon MQ ブローカーエンジンバージョンのアップグレード • Spring XML 設定ファイルの使用
2022 年 7 月 14 日	Amazon MQ が、マイナーエンジンバージョンのリリースである ActiveMQ 5.16.5 をサポートするようになりました。詳細については次を参照してください: • ActiveMQ 5.16.5 リリースページ • Amazon MQ for ActiveMQ エンジンバージョンの管理 • Spring XML 設定ファイルの使用 • Amazon MQ ブローカーエンジンバージョンのアップグレード
2022 年 5 月 4 日	Amazon MQ は、ブローカー設定の `networkConnector` 要素に包括的な言語を追加します。 • ブローカーの Amazon MQ ネットワークの作成と設定

日付	ドキュメントの更新
2022 年 4 月 25 日	Amazon MQ このリリースでは、`CRITICAL_ACTION_REQUIRED` ブローカーステートと `ActionRequired` API プロパティを追加します。`CRITICAL_ACTION_REQUIRED` は、ブローカーが低下したときに通知します。`ActionRequired` には、デベロッパーガイドで問題の解決方法を見つけるために使用するコードが用意されています。 • トラブルシューティング • Amazon MQ API リファレンス 内の `ActionRequired` ドキュメント。
2022 年 4 月 20 日	Amazon MQ が、新しいマイナーエンジンバージョンのリリースである ActiveMQ 5.16.4 をサポートするようになりました。詳細については次を参照してください: • ActiveMQ 5.16.4 Release ページ • Amazon MQ for ActiveMQ エンジンバージョンの管理 • Spring XML 設定ファイルの使用 • Amazon MQ ブローカーエンジンバージョンのアップグレード
2022 年 3 月 1 日	Amazon MQ がアジアパシフィック (ジャカルタ) リージョンで利用可能になりました。利用可能なリージョンの詳細については、AWS 全般リファレンスガイドの「AWS リージョンとエンドポイント」を参照してください。
2022 年 2 月 25 日	Amazon MQ for RabbitMQ が RabbitMQ バージョン 3.8.27 をサポートするようになりました。 このリリースでの修正と機能の詳細については、以下を参照してください。 • RabbitMQ サーバー GitHub リポジトリの RabbitMQ 3.8.27 リリースノート • RabbitMQ changelog サポートされる Amazon MQ for RabbitMQ のバージョンとブローカーアップグレードの詳細については、「Amazon MQ for RabbitMQ エンジンバージョンの管理」を参照してください。

日付	ドキュメントの更新
2022 年 2 月 16 日	Amazon MQ が アフリカ (ケープタウン) リージョン で利用可能になりました。利用可能なリージョンの詳細については、AWS 全般リファレンスガイドの「AWS リージョンとエンドポイント」を参照してください。
2022 年 2 月 14 日	Amazon MQ for RabbitMQ が RabbitMQ version 3.9.13 をサポートするようになりました。マイナーバージョンの自動アップグレードは、Rabbit 3.8 から 3.9 へのアップグレードには使用できません。これを行うには、ブローカーを手動でアップグレードします。 RabbitMQ 3.9 で導入された新機能の詳細については、GitHub ウェブサイトのバージョン 3.9.0 のリリースノートページを参照してください。 **Note** 現在、Amazon MQ はストリーム、または RabbitMQ 3.9 で導入された JSON での構造化ロギングの使用はサポートしません。 このリリースでの修正と機能の詳細については、以下を参照してください。 • RabbitMQ サーバー GitHub リポジトリの RabbitMQ 3.9.13 リリースノート • RabbitMQ changelog サポートされる Amazon MQ for RabbitMQ のバージョンとブローカーアップグレードの詳細については、「Amazon MQ for RabbitMQ エンジンバージョンの管理」を参照してください。
2022 年 2 月 7 日	Amazon MQ for RabbitMQ では、新しいブローカーメトリクスが導入され、クラスターデプロイの 3 つのノードすべてで平均リソース使用率を監視できます。 詳細については次を参照してください: • the section called “RabbitMQ のメトリクス”

日付	ドキュメントの更新
2022 年 1 月 18 日	Amazon MQ for RabbitMQ が RabbitMQ バージョン 3.8.26 をサポートするようになりました。 このリリースでの修正と機能の詳細については、以下を参照してください。 • RabbitMQ サーバー GitHub リポジトリの RabbitMQ 3.8.26 リリースノート • RabbitMQ changelog サポートされる Amazon MQ for RabbitMQ のバージョンとブローカーアップグレードの詳細については、「Amazon MQ for RabbitMQ エンジンバージョンの管理」を参照してください。
2022 年 1 月 13 日	Amazon MQ では、ブローカーが高メモリアラームを発して異常な状態にあるときに通知するための RABBITMQ_MEMORY_ALARM ステータスコードが導入されました。Amazon MQ では、高メモリアラームの診断、解決、および防止に役立つ詳細情報と推奨事項が提供されています。詳細については、以下を参照してください。 • the section called “ RABBITMQ_MEMORY_ALARM ”
2022 年 1 月 6 日	Amazon MQ for ActiveMQ ブローカーの CloudWatch Logs を設定すると、Amazon MQ が IAM リソースベースのポリシーの aws:SourceArn および aws:SourceAccount グローバル条件コンテキストキーを使用して「混乱した代理」問題の防止をサポートします。詳細については、以下を参照してください。 • the section called “サービス間での不分別な代理処理の防止”
2021 年 12 月 20 日	Amazon MQ for ActiveMQ では、一連の新しいメトリクスが導入され、サポートされている各種トランスポートプロトコルを使用してブローカーに接続できる最大数をモニタリングできるようになりました。また、ブローカーのネットワークでブローカーに接続されているノードの数をモニタリングできる追加の新しいメトリクスも導入されています。詳細については、以下を参照してください。 • the section called “ActiveMQ のメトリクス”

日付	ドキュメントの更新
2021 年 11 月 16 日	Amazon MQ for RabbitMQ が RabbitMQ バージョン 3.8.23 をサポートするようになりました。 このリリースでの修正と機能の詳細については、以下を参照してください。 • RabbitMQ サーバー GitHub リポジトリの RabbitMQ 3.8.23 リリースノート • RabbitMQ changelog サポートされる Amazon MQ for RabbitMQ のバージョンとブローカーアップグレードの詳細については、「Amazon MQ for RabbitMQ エンジンバージョンの管理」を参照してください。
2021 年 10 月 12 日	Amazon MQ が、新しいマイナーエンジンバージョンのリリースである ActiveMQ 5.16.3 をサポートするようになりました。詳細については次を参照してください: • ActiveMQ 5.16.3 Release ページ • Amazon MQ for ActiveMQ エンジンバージョンの管理 • Amazon MQ ブローカーエンジンバージョンのアップグレード • Spring XML 設定ファイルの使用

日付	ドキュメントの更新
2021 年 9 月 8 日	Amazon MQ for RabbitMQ が RabbitMQ バージョン 3.8.22 をサポートするようになりました。 このリリースには、以前にサポートされていたバージョンの RabbitMQ 3.8.17 で特定された、メッセージごとの TTL (有効期限) を使用するキューの問題に対する修正が含まれます。既存のブローカーをバージョン 3.8.22 にアップグレードすることをお勧めします。 このリリースでの修正と機能の詳細については、以下を参照してください。 • RabbitMQ サーバー GitHub リポジトリの RabbitMQ 3.8.22 リリースノート • RabbitMQ changelog サポートされる Amazon MQ for RabbitMQ のバージョンとブローカーアップグレードの詳細については、「Amazon MQ for RabbitMQ エンジンバージョンの管理」を参照してください。
2021 年 8 月 25 日	Amazon MQ for RabbitMQ は、メッセージごとの有効期限 (TTL) を使用するキューで特定された問題のため、RabbitMQ エンジンバージョン 3.8.17 を一時的に無効化しました。バージョン 3.8.11 の使用をお勧めします。
2021 年 7 月 29 日	Amazon MQ for RabbitMQ が RabbitMQ バージョン 3.8.17 をサポートするようになりました。この更新に含まれる修正と機能の詳細については、以下を参照してください。 • RabbitMQ サーバー GitHub リポジトリの RabbitMQ 3.8.17 リリースノート • RabbitMQ changelog • Amazon MQ for RabbitMQ エンジンバージョンの管理
2021 年 7 月 16 日	Amazon MQ ブローカーのメンテナンスウィンドウは AWS マネジメントコンソール、、 AWS CLI、または Amazon MQ API を使用して調整できるようになりました。ブローカーのメンテナンスウィンドウの詳細については、以下を参照してください。 • Amazon MQ ブローカーのメンテナンスウィンドウのスケジュール

日付	ドキュメントの更新
2021 年 7 月 6 日	Amazon MQ for RabbitMQ がコンシステントハッシュエクスチェンジタイプのサポートを導入しました。コンシステントハッシュエクスチェンジは、メッセージのルーティングキーから計算されたハッシュ値に基づいてメッセージをキューに送信します。詳細については次を参照してください: • コンシステントハッシュエクスチェンジプラグイン • RabbitMQ GitHub リポジトリの RabbitMQ Consistent Hash Exchange Type
2021 年 6 月 7 日	Amazon MQ が、新しいメジャーエンジンバージョンのリリースである ActiveMQ 5.16.2 をサポートするようになりました。詳細については次を参照してください: • ActiveMQ 5.16.2 Release ページ • Amazon MQ for ActiveMQ エンジンバージョンの管理 • Amazon MQ ブローカーエンジンバージョンのアップグレード • Spring XML 設定ファイルの使用
2021 年 5 月 26 日	Amazon MQ for RabbitMQ が、中国 (北京) および中国 (寧夏) リージョンで利用可能になりました。利用可能なリージョンについては、AWS のリージョンとエンドポイントを参照してください。
2021 年 5 月 18 日	Amazon MQ for RabbitMQ がブローカーデフォルトを実装します。 ブローカーを初めて作成するときは、ブローカーのパフォーマンスを最適化するために、Amazon MQ が選択されたインスタンスタイプとブローカーデプロイモードに基づいて一連のブローカーポリシーと vhost 制限を作成します。詳細については、以下を参照してください。 https://docs.aws.amazon.com//amazon-mq/latest/developer-guide/rabbitmq-defaults.html

日付	ドキュメントの更新
2021 年 5 月 5 日	Amazon MQ が ActiveMQ 5.15.15 をサポートするようになりました。詳細については次を参照してください: • ActiveMQ 5.15.15 Release ページ • Amazon MQ for ActiveMQ エンジンバージョンの管理 • Spring XML 設定ファイルの使用
2021 年 5 月 5 日	Amazon MQ は、 AWS 管理ポリシーの変更の追跡を開始しました。詳細については次を参照してください: • the section called “AWS マネージドポリシー”
2021 年 4 月 14 日	Amazon MQ が中国 (北京) および中国 (寧夏) リージョンで利用可能になりました。利用可能なリージョンについては、AWS のリージョンとエンドポイントを参照してください。
2021 年 4 月 7 日	Amazon MQ が RabbitMQ 3.8.11 をサポートするようになりました。この更新に含まれる修正と機能の詳細については、以下を参照してください。 • RabbitMQ サーバー GitHub リポジトリの RabbitMQ 3.8.11 リリースノート • RabbitMQ changelog • Amazon MQ for RabbitMQ エンジンバージョンの管理
2021 年 4 月 1 日	Amazon MQ がアジアパシフィック (大阪) リージョンで利用可能になりました。利用可能なリージョンについては、Amazon MQ のリージョンとエンドポイントを参照してください。

日付	ドキュメントの更新
2020 年 12 月 21 日	Amazon MQ が ActiveMQ 5.15.14 をサポートするようになりました。詳細については次を参照してください: • ActiveMQ 5.15.14 リリースノート • Amazon MQ for ActiveMQ エンジンバージョンの管理 • Spring XML 設定ファイルの使用 • **Important** このリリースでの既知の Apache ActiveMQ 問題のため、Amazon MQ for ActiveMQ では、ActiveMQ ウェブコンソールの新しい [Pause Queue] ボタンを使用できません。この問題の詳細については、「AMQ-8104」を参照してください。

日付	ドキュメントの更新
2020 年 11 月 4 日	Amazon MQ が、人気のあるオープンソースのメッセージブローカーである RabbitMQ をサポートするようになりました。これにより、コードを書き換え AWS ることなく、既存の RabbitMQ メッセージブローカーを に移行できます。 Amazon MQ for RabbitMQ は、個々のメッセージブローカーとクラスター化されたメッセージブローカーの両方を管理し、インフラストラクチャのプロビジョニング、ブローカーのセットアップ、およびソフトウェアの更新などのタスクを処理します。 • Amazon MQ は RabbitMQ 3.8.6 をサポートします。サポートされるエンジンバージョンの詳細については、「the section called “バージョン管理”」を参照してください。 • AWS 無料利用枠には、1 年間毎月最大 750 時間の単一インスタンス `mq.t3.micro` ブローカーと、最大 20GB のストレージが含まれています。サポートされているインスタンスタイプの詳細については、「Broker instance types」を参照してください。 • Amazon MQ for RabbitMQ では、AMQP 0-9-1、および RabbitMQ クライアントライブラリでサポートされる任意の言語を使用してブローカーにアクセスできます。サポートされるプロトコルと暗号化スイートの詳細については、「the section called “Amazon MQ for RabbitMQ のプロトコル”」を参照してください。 • RabbitMQ for Amazon MQ は、現在 Amazon MQ を利用できるすべてのリージョンでご利用いただけます。利用可能なすべてのリージョンの詳細については、「AWS リージョン表」を参照してください。 Amazon MQ の使用を開始し、ブローカーを作成して、JVM ベースのアプリケーションを RabbitMQ ブローカーに接続するには、「開始方法: RabbitMQ ブローカーの作成と接続」を参照してください。

日付	ドキュメントの更新
2020 年 10 月 22 日	Amazon MQ は ActiveMQ 5.15.13 をサポートします。詳細については次を参照してください: • ActiveMQ 5.15.13 リリースノート • Amazon MQ for ActiveMQ エンジンバージョンの管理 • Spring XML 設定ファイルの使用
2020 年 9 月 30 日	Amazon MQ が欧州 (ミラノ) リージョンで利用可能になりました。利用可能なリージョンについては、Amazon MQ のリージョンとエンドポイントを参照してください。
2020 年 7 月 27 日	Amazon MQ ユーザーは、アクティブディレクトリまたはその他の LDAP サーバーに保存されている認証情報を使用して認証することができます。Amazon MQ ユーザーの追加、削除、変更、およびトピックとキューへの許可の割り当てを行うことも可能です。詳細については、「LDAP を ActiveMQ に統合する」を参照してください。
2020 年 7 月 17 日	Amazon MQ が `mq.t3.micro` インスタンスタイプをサポートするようになりました。詳細については、「Broker instance types」を参照してください。
2020 年 6 月 30 日	Amazon MQ は ActiveMQ 5.15.12 をサポートします。詳細については次を参照してください: • ActiveMQ 5.15.12 リリースノート • Amazon MQ for ActiveMQ エンジンバージョンの管理 • Spring XML 設定ファイルの使用

日付	ドキュメントの更新
2020 年 4 月 30 日	Amazon MQ は、broker 要素の新しい子コレクション要素 systemUsage をサポートしています。詳細については、「systemUsage」を参照してください。 Amazon MQ は、kahaDB 子要素の 3 つの新しい属性もサポートします。 • journalDiskSyncInterval - の場合にディスク同期を実行する間隔 (ミリ秒) journalDiskSyncStrategy=periodic 。 • journalDiskSyncStrategy - ディスク同期ポリシーを設定します。 • preallocationStrategy - 新しいジャーナルファイルが必要になったときにブローカーがジャーナルファイルの事前割り当てを試みる方法を設定します。 詳細については、「属性」を参照してください。
2020 年 3 月 3 日	Amazon MQ が 2 つの新しい CloudWatch メトリクスをサポート • TempPercentUsage - 非永続的メッセージで使用可能な一時ストレージの割合 (%)。 • JobSchedulerStorePercentUsage - ジョブスケジューラストアで使用するディスク領域の割合 (%)。 詳細については、「Monitoring and logging Amazon MQ brokers」を参照してください。
2020 年 2 月 4 日	Amazon MQ をアジアパシフィック (香港) および中東 (バーレーン) リージョンでご利用いただけます。利用可能なリージョンについては、AWS のリージョンとエンドポイントを参照してください。
2020 年 1 月 22 日	Amazon MQ は ActiveMQ 5.15.10 をサポートします。詳細については次を参照してください: • ActiveMQ 5.15.10 リリースノート • Amazon MQ for ActiveMQ エンジンバージョンの管理 • Spring XML 設定ファイルの使用

日付	ドキュメントの更新
2019 年 12 月 19 日	Amazon MQ を欧州 (ストックホルム) および南米 (サンパウロ) リージョンでご利用いただけます。利用可能なリージョンについては、AWS のリージョンとエンドポイントを参照してください。
2019 年 12 月 16 日	Amazon MQ は、デフォルトの Amazon Elastic File System (Amazon EFS) ではなく、ブローカーストレージ用の Amazon Elastic Block Store (EBS) を使用することによるスループット最適化ブローカーの作成をサポートします。複数のアベイラビリティーゾーン全体で優れた耐障害性とレプリケーションを活用するには、Amazon EFS を使用します。低レイテンシーと高スループットを活用するには、Amazon EBS を使用します。 **Important** • Amazon EBS を使用できるのは、`mq.m5` ブローカーインスタンスタイプファミリーのみです。 • ブローカーインスタンスタイプを変更することはできますが、ブローカーを作成した後でブローカーストレージタイプを変更することはできません。 • Amazon EBS は単一のアベイラビリティーゾーン内でデータをレプリケートし、ActiveMQ アクティブ/スタンバイデプロイモードをサポートしません。 詳細については次を参照してください: • Storage • 最高のスループットのために正しいブローカーストレージタイプを選択する • Amazon MQ REST API リファレンスの `broker-instance-options` リソースの `storageType` プロパティ • Monitoring and logging Amazon MQ brokers セクションの `BurstBalance` 、`VolumeReadOps` 、および `VolumeWriteOps` メトリクス。

日付	ドキュメントの更新
2019 年 10 月 18 日	`TotalEnqueueCount` および `TotalDequeueCount` の 2 つの Amazon CloudWatch メトリクスをご利用いただけます。詳細については、「Monitoring and logging Amazon MQ brokers」を参照してください。
2019 年 10 月 11 日	Amazon MQ が、米国商用リージョンで米国連邦情報処理規格 140-2 (FIPS) 準拠のエンドポイントをサポートするようになりました。 詳細については、以下を参照してください。 • 連邦情報処理規格 (FIPS) 140-2 • Amazon MQ のリージョンとエンドポイント
2019 年 9 月 30 日	Amazon MQ に、ホストインスタンスタイプを変更してブローカーをスケーリングする機能が組み込まれました。詳細については、`UpdateBrokerInput` の `hostInstanceType` プロパティおよび `DescribeBrokerOutput` の `pendingHostInstanceType` プロパティを参照してください。
2019 年 8 月 30 日	コンソールと `UpdateBrokerInput` の両方で、ブローカーに関連付けられたセキュリティグループを更新できるようになりました。
2019 年 7 月 22 日	Amazon MQ は AWS Key Management Service (KMS) と統合して、サーバー側の暗号化を提供します。独自のカスタマーマネージド CMK を選択するか、 AWS KMS アカウントで AWS マネージド KMS キーを使用できるようになりました。詳細については、「保管中の暗号化」を参照してください。 Amazon MQ は、次の方法で AWS KMS キーの使用をサポートしています。 • AWS 所有の KMS キー — キーは Amazon MQ を所有しており、アカウント内にはありません。 • AWS マネージド KMS キー — AWS マネージド KMS キー (aws/mq) は、Amazon MQ によってユーザーに代わって作成、管理、使用されるアカウントの KMS キーです。 • 既存のカスタマーマネージド CMK を選択する – カスタマーマネージド CMK は、 AWS Key Management Service (KMS) でユーザーが作成し、管理します。

日付	ドキュメントの更新
2019 年 6 月 19 日	Amazon MQ を欧州 (パリ) およびアジアパシフィック (ムンバイ) リージョンでご利用いただけます。利用可能なリージョンについては、AWS のリージョンとエンドポイントを参照してください。
2019 年 6 月 12 日	Amazon MQ をカナダ (中部) リージョンでご利用いただけます。利用可能なリージョンについては、AWS のリージョンとエンドポイントを参照してください。
2019 年 6 月 3 日	`EstablishedConnectionsCount` および `InactiveDurableSubscribers` の 2 つの新しい Amazon CloudWatch メトリクスをご利用いただけます。詳細については次を参照してください: • Monitoring and logging Amazon MQ brokers • Monitoring and logging Amazon MQ brokers
2019 年 5 月 10 日	新しい `mq.t2.micro` インスタンスタイプのデータストレージが 20 GB に制限されました。詳細については次を参照してください: • the section called “データストレージ” • Broker instance types
2019 年 4 月 29 日	タグベースのポリシーとリソースレベルのアクセス権限を使用できるようになりました。詳細については次を参照してください: • Amazon MQ で IAM が機能する仕組み • Amazon MQ API アクションに対するリソースレベルの許可
2019 年 4 月 16 日	REST API を使用して、ブローカーエンジンとブローカーインスタンスのオプションに関する情報を取得できるようになりました。詳細については次を参照してください: • ブローカーインスタンスのオプション • ブローカーエンジンタイプ

日付	ドキュメントの更新
2019 年 4 月 8 日	Amazon MQ は ActiveMQ 5.15.9 をサポートします。詳細については次を参照してください: • ActiveMQ 5.15.9 リリースノート • Amazon MQ for ActiveMQ エンジンバージョンの管理 • Spring XML 設定ファイルの使用
2019 年 3 月 4 日	動的なフェイルオーバーの設定と、ブローカーのネットワークのクライアントの再分散のため、ドキュメントを改善しました。`transportConnectors` と `networkConnectors` 設定オプションを設定することにより、動的なフェイルオーバーを有効にします。詳細については次を参照してください: • トランスポートコネクタを使用した動的なフェイルオーバー • Amazon MQ のブローカーのネットワーク • Amazon MQ Broker Configuration Parameters
2019 年 2 月 27 日	Amazon MQ は、以下のリージョンに加えて、欧州 (ロンドン) リージョンでもご利用いただけます。 • アジアパシフィック (シンガポール) • 米国東部(オハイオ) • 米国東部 (バージニア北部) • 米国西部 (北カリフォルニア) • 米国西部 (オレゴン) • アジアパシフィック (東京) • アジアパシフィック (ソウル) • アジアパシフィック (シドニー) • 欧州 (フランクフルト) • 欧州 (アイルランド)
2019 年 1 月 24 日	デフォルト設定に、非アクティブな送信先を消去するポリシーが含まれるようになりました。

日付	ドキュメントの更新
2019 年 1 月 17 日	Amazon MQ `mq.t2.micro` インスタンスタイプが、ワイヤレベルプロトコルあたり 100 個の接続のみをサポートするようになりました。詳細については、「Quotas in Amazon MQ」を参照してください。
2018 年 12 月 19 日	ブローカーのネットワークで一連の Amazon MQ ブローカーを設定できます。詳細については、次のセクションを参照してください。 • Amazon MQ のブローカーのネットワーク • Creating and Configuring a Network of Brokers • ブローカーのネットワークを正しく設定する • `networkConnector` • `networkConnectionStartAsync`

2018 年 12 月 11 日	Amazon MQ は ActiveMQ 5.15.8、5.15.6、および 5.15.0 をサポートします。
	• 解決されたバグと ActiveMQ の改善点。
	  • ActiveMQ 5.15.8 リリースノート
	  • ActiveMQ 5.15.7 リリースノート

2018 年 12 月 5 日	AWS は、コスト配分の追跡に役立つリソースのタグ付けをサポートしています。リソースを作成するとき、またはそのリソースの詳細を表示することによって、リソースにタグを付けることができます。詳細については、「リソースにタグを付ける」を参照してください。
2018 年 11 月 19 日	AWS は、SOC コンプライアンスプログラムを拡張し、SOC 準拠サービスとして Amazon MQ を含めました。
2018 年 10 月 15 日	• ユーザーあたりのグループの最大数は 20 です。詳細については、「Users」を参照してください。 • 接続の最大数は、ブローカーあたり、ワイヤレベルプロトコルあたり 1,000 です。詳細については、「ブローカー」を参照してください。
2018 年 10 月 2 日	AWS は、HIPAA コンプライアンスプログラムを拡張し、HIPAA 対応サービスとして Amazon MQ を含めました。

日付	ドキュメントの更新

2018 年 9 月 27 日	ActiveMQ 5.15.0 に加えて、Amazon MQ が 5.15.6 をサポートします。詳細については次を参照してください:
	• 開始方法: ActiveMQ ブローカーの作成と接続
	• 解決されたバグと ActiveMQ ドキュメントの改善点。
	  • ActiveMQ 5.15.6 リリースノート
	  • ActiveMQ 5.15.5 リリースノート
	  • ActiveMQ 5.15.4 リリースノート
	  • ActiveMQ 5.15.3 リリースノート
	  • ActiveMQ 5.15.2 リリースノート
	  • ActiveMQ 5.15.1 リリースノート
	• ActiveMQ Client 5.15.6

2018 年 8 月 31 日	• 以下のメトリクスが利用可能です。
	  • `CurrentConnectionsCount`
	  • `TotalConsumerCount`
	  • `TotalProducerCount`
	  詳細については「Monitoring and logging Amazon MQ brokers」セクションを参照してください。
	• また、ブローカーの IP アドレスが [詳細] ページに表示されます。
	  Note
	  パブリックアクセシビリティが無効なブローカーの場合、内部 IP アドレスが表示されます。

日付	ドキュメントの更新
2018 年 8 月 30 日	Amazon MQ は、以下のリージョンに加えて、アジアパシフィック (シンガポール) リージョンでもご利用いただけます。 • 米国東部(オハイオ) • 米国東部 (バージニア北部) • 米国西部 (北カリフォルニア) • 米国西部 (オレゴン) • アジアパシフィック (東京) • アジアパシフィック (ソウル) • アジアパシフィック (シドニー) • 欧州 (フランクフルト) • 欧州 (アイルランド)
2018 年 7 月 30 日	一般ログと監査ログを Amazon CloudWatch Logs に発行するように Amazon MQ を設定できます。詳細については、「Monitoring and logging Amazon MQ brokers」を参照してください。
2018 年 7 月 25 日	Amazon MQ は、以下のリージョンに加えて、アジアパシフィック (東京) およびアジアパシフィック (ソウル) リージョンでもご利用いただけます。 • 米国東部(オハイオ) • 米国東部 (バージニア北部) • 米国西部 (北カリフォルニア) • 米国西部 (オレゴン) • アジアパシフィック (シドニー) • 欧州 (フランクフルト) • 欧州 (アイルランド)
2018 年 7 月 19 日	を使用して Amazon MQ API コール AWS CloudTrail を記録できます。詳細については、「Logging Amazon MQ API calls using CloudTrail」を参照してください。

日付	ドキュメントの更新
2018 年 6 月 29 日	mq.t2.micro および mq.m4.large に加えて、次のブローカーインスタンスタイプが一般的な開発、テスト、および高度なスループットが必要なプロダクションワークロードに利用できます。 • mq.m5.large • mq.m5.xlarge • mq.m5.2xlarge • mq.m5.4xlarge 詳細については、「Broker instance types」を参照してください。
2018 年 6 月 27 日	Amazon MQ は、以下のリージョンに加えて、米国西部 (北カリフォルニア) リージョンでもご利用いただけます。 • 米国東部(オハイオ) • 米国東部 (バージニア北部) • 米国西部 (オレゴン) • アジアパシフィック (シドニー) • 欧州 (フランクフルト) • 欧州 (アイルランド)

日付	ドキュメントの更新

2018 年 6 月 14 日	• AWS::Amazon MQ::Broker AWS CloudFormation リソースを使用して、次のアクションを実行できます。
	  • ブローカーの作成。
	  • 指定されたブローカーの設定の変更またはユーザーの変更。
	  • 指定されたブローカーに関する情報の戻し。
	  • 指定されたブローカーの削除。
	
	  Note
	  Amazon MQ の Broker ConfigurationId または Amazon MQ の Broker User プロパティタイプのプロパティを変更すると、ブローカーは直ちに再起動されます。
	• AWS::Amazon MQ::Configuration AWS CloudFormation リソースを使用して、次のアクションを実行できます。
	  • 設定の作成。
	  • 指定された構成の更新。
	  • 指定された設定に関する情報の戻し。
	
	  Note
	  を使用して CloudFormation 、Amazon MQ 設定を変更できますが、削除はできません。

2018 年 6 月 7 日	Amazon MQ コンソールは、ドイツ語、ポルトガル語 (ブラジル)、スペイン語、イタリア語、および繁体字中国語をサポートします。
2018 年 5 月 17 日	ブローカーあたりのユーザー数の制限は 250 です。詳細については、「Users」を参照してください。
2018 年 3 月 13 日	ブローカーの作成には約 15 分かかります。詳細については、「ブローカー作成の完了」を参照してください。

日付	ドキュメントの更新
2018 年 3 月 1 日	• ??? 属性を使用して Apache KahaDB のconcurrentStoreAnd DispatchQueues 同時保存とディスパッチを設定できます。 • mq.t2.micro ブローカーインスタンスタイプに CpuCreditBalance CloudWatch メトリクスを利用できます。
2018 年 1 月 10 日	以下の変更は Amazon MQ コンソールに影響を及ぼします。 • ブローカーのリストで、[Creation (作成)] 列はデフォルトで非表示になります。ページサイズと列をカスタマイズするには、  。 • ***[MyBroker]*** ページの [接続] セクションでセキュリティグループの名前または  を選択すると、(VPC コンソールではなく) EC2 コンソールが開きます。EC2 コンソールでは、インバウンドおよびアウトバウンドルールのより直感的な設定ができます。詳細については、更新された「Connecting a Java application to your broker」セクションを参照してください。
2018 年 1 月 9 日	• REST オペレーション ID UpdateBroker の許可が、IAM コンソールで mq:UpdateBroker として正しく表示されるようになりました。 • 誤った mq:DescribeEngine 許可は IAM コンソールから削除されました。

日付	ドキュメントの更新

2017 年 11 月 28 日	これは、Amazon MQ と Amazon MQ デベロッパーガイドの初回リリースです。
	• Amazon MQ は、以下のリージョンでご利用いただけます。
	  • 米国東部(オハイオ)
	  • 米国東部 (バージニア北部)
	  • 米国西部 (オレゴン)
	  • アジアパシフィック (シドニー)
	  • 欧州 (フランクフルト)
	  • 欧州 (アイルランド)
	  `mq.t2.micro` インスタンスタイプの使用は CPU クレジットとベースラインパフォーマンスの対象となり、ベースラインレベルを超えてバーストする機能を備えています (詳細については、「`CpuCreditBalance` メトリクス」を参照してください)。アプリケーションに一定のパフォーマンスが必要な場合は、`mq.m5.large` インスタンスタイプの使用を検討してください。
	• `mq.m4.large` および `mq.t2.micro` ブローカーを作成できます。
	  `mq.t2.micro` インスタンスタイプの使用は CPU クレジットとベースラインパフォーマンスの対象となり、ベースラインレベルを超えてバーストする機能を備えています (詳細については、「`CpuCreditBalance` メトリクス」を参照してください)。アプリケーションに一定のパフォーマンスが必要な場合は、`mq.m5.large` インスタンスタイプの使用を検討してください。
	• ActiveMQ 5.15.0 ブローカーエンジンを使用できます。
	• Amazon MQ REST API と AWS SDKs を使用して、プログラムでブローカーを作成および管理することもできます。
	• ブローカーには、ActiveMQ がサポートする任意のプログラミング言語を使用し、以下のプロトコルに対して TLS を明示的に有効にすることによってアクセスできます。
	  • AMQP
	  • MQTT

日付	ドキュメントの更新
	• MQTT over WebSocket • OpenWire • STOMP • STOMP over WebSocket • ActiveMQ ブローカーには、さまざまな ActiveMQ クライアントを使用して接続できます。ActiveMQ クライアントを使用することをお勧めします。詳細については、「Connecting a Java application to your broker」を参照してください。 • ブローカーは任意のサイズのメッセージを送受信できます。

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。