aws

開発者ガイド

Amazon MQ

Amazon MQ: 開発者ガイド

Table of Contents

Amazon MQ とは 1
Amazon MQ は Amazon SQS や Amazon SNS とどのような点で異なりますか。 1
Amazon MQ の使用を開始するにはどうすればよいですか。 1
ご意見をお待ちしております 2
設定 3
ステップ 1: 前提条件 3
にサインアップする AWS アカウント 3
管理アクセスを持つユーザーを作成する 4
ユーザーを作成して AWS 認証情報を取得する 5
ステップ 3: サンプルコードの使用準備を整える 6
次のステップ 7
開始方法 8
前提条件 8
ActiveMQ ブローカーの作成と接続 8
ステップ 1: ActiveMQ ブローカーを作成する 9
ステップ 2: ブローカーに Java アプリケーションを接続する 11
ステップ 3: (オプション) AWS Lambda 関数に接続する 17
ステップ 4: ブローカーを削除する 19
次のステップ 19
RabbitMQ ブローカーの作成と接続 20
ステップ 1: RabbitMQ ブローカーを作成する 20
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 23
ステップ 3: (オプション) AWS Lambda 関数に接続する 27
ステップ 4: ブローカーを削除する 30
次のステップ 30
ブローカーの管理 32
ブローカーのメンテナンス 32
ブローカーメンテナンスウィンドウの調整 33
エンジンバージョンのアップグレード 36
エンジンバージョンの手動アップグレード 37
マイナーエンジンバージョンの自動アップグレード 40
エンジンバージョンのサポート終了カレンダー 42
ブローカーステータス 42
ブローカーのリスト化とブローカー詳細の表示 43

ブローカーをリストしてブローカーの詳細を表示する 43
パブリックアクセシビリティが無効化されたブローカーウェブコンソールへのアクセス 46
前提条件 46
パブリックアクセシビリティが無効化されているブローカーのウェブコンソールにアクセスする 47
ブローカーの再起動 47
Amazon MQ ブローカーを再起動する 48
ブローカーの削除 48
Amazon MQ ブローカーの削除 49
ブローカーの設定 49
ブローカー設定のライフサイクル 49
インスタンスのタイプ 50
Amazon MQ for ActiveMQ インスタンスタイプ 51
Amazon MQ for RabbitMQ インスタンスタイプ 52
リソースのタグ付け 53
コスト割り当てのタグ付け 53
Amazon MQ コンソールでのタグの管理 54
Amazon MQ API アクションを使用した管理 55
Amazon MQ for ActiveMQ 56
ActiveMQ エンジン 56
基本的要素 57
ブローカーのアーキテクチャ 69
ブローカーの設定 83
バージョン管理 118
Java の実用例 119
ActiveMQ チュートリアル 130
ブローカーの作成と設定 131
ブローカーのネットワークの作成と設定 138
ブローカーへの Java アプリケーションの接続 144
ActiveMQ ブローカーの LDAP との統合 150
ブローカーユーザーの作成と管理 165
Amazon MQ for ActiveMQ のベストプラクティス 168
Amazon MQ への接続 168
効果的な Amazon MQ パフォーマンスの確保 171
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 174
クロスリージョンデータレプリケーション 176

プライマリブローカーとレプリカブローカー ... 176
CRDR ブローカーの作成/削除 ... 177
スイッチオーバー/フェイルオーバーの開始 ... 182
メトリクス ... 185
クォータ ... 187
ブローカー ... 187
Configurations ... 188
[ユーザー] ... 189
データストレージ ... 190
API スロットリング ... 191
Amazon MQ for RabbitMQ ... 192
RabbitMQ エンジン ... 192
基本的要素 ... 192
ブローカーのアーキテクチャ ... 212
ブローカーの設定 ... 215
バージョン管理 ... 220
RabbitMQ のチュートリアル ... 222
ブローカー設定の編集 ... 223
Amazon MQ for RabbitMQ でPython Pika を使う ... 224
一時停止されたキュー同期の解決 ... 231
Amazon MQ for RabbitMQ のベストプラクティス ... 237
レイジーキューを有効にする ... 238
永続キューと持続キューを使用する ... 239
キューを短くしておく ... 239
承認と確認を設定する ... 240
プリフェッチを設定する ... 241
Celery を設定 ... 242
ネットワーク障害から自動的に回復する ... 243
RabbitMQ ブローカーの Classic Queue v2 を有効にする ... 243
クォータ ... 244
ブローカー ... 245
データストレージ ... 245
API スロットリング ... 246
セキュリティ ... 247
データ保護 ... 248
暗号化 ... 249

保管中の暗号化 249
転送中の暗号化 258
ID とアクセス管理 260
対象者 260
アイデンティティを使用した認証 261
ポリシーを使用したアクセスの管理 264
Amazon MQ で IAM が機能する仕組み 267
アイデンティティベースポリシーの例 273
API 認証と認可 276
AWS マネージドポリシー 280
サービスリンクロールの使用 281
トラブルシューティング 287
コンプライアンス検証 289
耐障害性 291
インフラストラクチャセキュリティ 291
セキュリティベストプラクティス 292
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 292
認可マップを常に設定する 292
不要なプロトコルをブロックする 292
ロギングとモニタリング 294
CloudWatch メトリクスへのアクセス 294
AWS Management Console 295
AWS Command Line Interface 297
Amazon CloudWatch API 297
CloudWatch を使用したブローカーのモニタリング 297
Amazon MQ for ActiveMQ ブローカーのロギングとモニタリング 298
Amazon MQ for RabbitMQ ブローカーのロギングとモニタリング 307
CloudTrailを使用したAPI呼び出しのログ記録 314
CloudTrail 内の Amazon MQ 情報 315
Amazon MQ ログファイルエントリの例 317
ログを CloudWatch Logs に発行するための Amazon MQ の設定 319
Amazon MQ for ActiveMQ ログの設定 319
Amazon MQ for RabbitMQ ログの設定 325
クォータ 326
ブローカー 326
Configurations 327

Users 328
データストレージ 329
API スロットリング 331
トラブルシューティング 332
トラブルシューティング: 一般 333
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。 333
SSL 例外 339
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。 339
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。 340
トラブルシューティング: Amazon MQ for ActiveMQ 341
CloudWatch ログの取得 341
再起動後にブローカーに接続する 342
一部のクライアントは接続できません 342
ウェブコンソールでの JSP 例外 343
トラブルシューティング: Amazon MQ の RabbitMQ 344
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。 344
Amazon MQ for 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? 344
ブローカーの Amazon VPC 設定を変更できません。 344
トラブルシューティング:Amazon MQ のアクションに必要なコード 345
RABBITMQ_MEMORY_ALARM 345
RABBITMQ_INVALID_KMS_KEY 352
BROKER_ENI_DELETED 353
BROKER_OOM 354
RABBITMQ_DISK_ALARM 355
関連リソース 358
Amazon MQ のリソース 358
Amazon MQ for ActiveMQ のリソース 359
Amazon MQ for RabbitMQ のリソース 359
リリースノート 361
ドキュメント履歴 394
AWS 用語集 409
..... cdx

Amazon MQ とは

Amazon MQ は、クラウド内のメッセージブローカーへの移行を容易にするマネージドメッセージブローカーサービスです。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。現在、Amazon MQ は Apache ActiveMQ Classic エンジンタイプと RabbitMQ エンジンタイプをサポートしています。

Amazon MQ は既存のアプリケーションおよびサービスと連動し、独自のメッセージングシステムを管理、運用、保守する必要はありません。

トピック

- Amazon MQ は Amazon SQS や Amazon SNS とどのような点で異なりますか。
- Amazon MQ の使用を開始するにはどうすればよいですか。
- ご意見をお待ちしております

Amazon MQ は Amazon SQS や Amazon SNS とどのような点で異なりますか。

Amazon MQ は、多くの一般的なメッセージブローカーとの互換性を提供するマネージドメッセージブローカーサービスです。JMS などの APIs や AMQP 0-9-1、AMQP 1.0、MQTT OpenWire、STOMP などのプロトコルとの互換性に依存する既存のメッセージブローカーからアプリケーションを移行するには、Amazon MQ をお勧めします。

Amazon SQS と Amazon SNS は、スケーラビリティに優れ、使いやすく、メッセージブローカーをセットアップする必要がないキューおよびトピックサービスです。これらのサービスは、ほぼスケーラビリティの拡張性とシンプルな API からメリットを得ることができる新規のアプリケーションに推奨されます。

Amazon MQ の使用を開始するにはどうすればよいですか。

- Amazon MQ で最初のブローカーを作成するには、「Getting Started with Amazon MQ」を参照してください。
- Amazon MQ を最大限に活用するためのガイドラインと注意事項については、「Working with Amazon MQ for ActiveMQ」と「Working with Amazon MQ for RabbitMQ」を参照してください。

- 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 コマンドの詳細については、「 コマンドAWS CLI リファレンス」の「Amazon MQ」を参照してください。

ご意見をお待ちしております

ご意見をお待ちしております。お問い合わせについては、Amazon MQ ディスカッションフォーラムにアクセスしてください。

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
- ステップ 2: ユーザーを作成して AWS 認証情報を取得する
- ステップ 3: サンプルコードの使用準備を整える
- 次のステップ

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、通話呼び出しを受け取り、電話キーパッドで検証コードを入力するように求められます。

   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。セキュリティのベストプラクティスとして、ユーザーに管理アクセスを割り当て、ルートユーザーのみを使用してルートユーザーアクセスが必要なタスクを実行してください。

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 [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、 を保護し AWS アカウントのルートユーザー、 を有効にして AWS IAM Identity Center、日常的なタスクにルートユーザーを使用しないように管理ユーザーを作成します。

のセキュリティ保護 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 Management Console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 「ルートユーザーの仮想 MFA デバイスを有効にする (コンソール) 」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 Identity Center ユーザーガイド」の「AWS IAM Identity Center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、「 ユーザーガイド」の「デフォルト でユーザーアクセス IAM アイデンティティセンターディレクトリを設定するAWS IAM Identity Center 」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 Identity Center ユーザーガイド」の「権限設定を作成する」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 Identity Center ユーザーガイド」の「グループの参加」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 外部で を操作する場合は、プログラムによるアクセスが必要です AWS Management Console。プログラムによるアクセスを許可する方法は、 に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マチック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。

プログラマチックアクセス権を必要とするユーザー	目的	方法
ワークフォースアイデンティティ (IAM Identity Center 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 SDKs AWS CLI、または AWS APIs。	使用するインターフェイス用の手引きに従ってください。 • については AWS CLI、「 ユーザーガイド」の AWS CLI 「 を使用するための の設定 AWS IAM Identity CenterAWS Command Line Interface 」を参照してください。 • AWS SDKs 、ツール、AWS APIs「 SDK とツールのリファレンスガイド」

プログラマチックアクセス権を必要とするユーザー	目的	方法
		の「IAM Identity Center 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 SDKs AWS CLI、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。
IAM	(非推奨) 長期認証情報を使用して、、 AWS SDKs AWS CLI、または AWS APIs。	使用するインターフェイス用の手引きに従ってください。 • 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 とツールのリファレンスガイド」の「長期的な認証情報を使用した認証」を参照してください。 AWS SDKs • AWS APIs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 Management Console 、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。

次のステップ

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。

Amazon MQ の開始方法

このセクションは、ActiveMQ または RabbitMQ ブローカー向けの Amazon MQ を作成する方法と、ブローカーにアプリケーションを接続する方法を説明することによって、Amazon MQ をより良く理解できるようにします。

ブローカーインスタンスの作成と接続は、ブローカーエンジンごとに若干異なります。ブローカーの作成と接続に関する詳しい情報については、以下のエンジンタイプから使用するエンジンを選択してください。ブローカーを作成して接続した後は、ブローカーを削除するために役立つ手順も記載されています。

トピック

- 前提条件
- ActiveMQ ブローカーの作成と接続
- RabbitMQ ブローカーの作成と接続

前提条件

開始する前に、「Setting Up Amazon MQ」の手順を完了してください。

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`、`t3`) およびサイズ (`large`、`micro`) を組み合わせた説明がブローカーインスタンスタイプ (`mq.m5.large` など) になります。詳細については、「ブローカー」を参照してください。

トピック

- ステップ 1: ActiveMQ ブローカーを作成する
- ステップ 2: ブローカーに Java アプリケーションを接続する
- ステップ 3: (オプション) AWS Lambda 関数に接続する
- ステップ 4: ブローカーを削除する
- 次のステップ

ステップ 1: ActiveMQ 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。次の例は、 を使用して基本的なブローカー AWS Management Console を作成する方法を示しています。

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、「Broker Architecture」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、「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、「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。
      - ブローカーのネットワークのサンプル設計図の詳細については、「サンプル設計図」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。ブローカー名には、 CloudWatch ログを含む他の 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。

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。

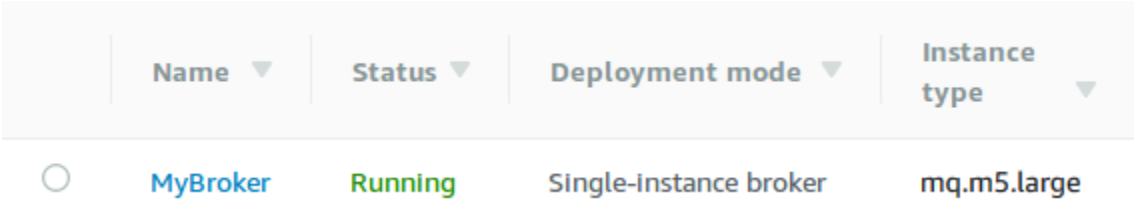
> **Important**
>
> 個人を特定できる情報 (PII) などの機密情報や秘匿性の高い情報はブローカーのユーザー名に追加しないでください。ブローカーのユーザー名は、 CloudWatch ログを含む他の AWS のサービスからアクセスできます。ブローカーのユーザー名は、プライベートデータや機密データとして使用することを意図していません。

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. を選択します***MyBroker***。

   MyBroker ページの Connect セクションで、ブローカーの ActiveMQ ウェブコンソール URL を書き留めます。次に例を示します。

   ```
   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。 OpenWire エンドポイントの例を次に示します。

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

ステップ 2: ブローカーに Java アプリケーションを接続する

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。

前提条件

VPC 属性 を有効にする

> **Note**
>
> 既存の Amazon MQ ブローカーのパブリックアクセシビリティを無効にすることはできません。

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. ブローカーリストから、ブローカーの名前を選択します (例: MyBroker) 。
3. ***MyBroker*** ページの Connections セクションで、ブローカーのウェブコンソール URL とワイヤレベルのプロトコルのアドレスとポートを書き留めます。
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

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、「Broker Architecture」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できます。

```
// 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

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();
```

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使用できます。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:CreateNetworkインターフェイス
   - ec2:DeleteNetworkインターフェイス
   - ec2:DescribeNetworkインターフェイス
   - ec2:DescribeSecurityグループ
   - ec2:DescribeSubnets
   - ec2:DescribeVpcs
   - ログ：CreateLogグループ
   - ログ：CreateLogストリーム
   - 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 Management Console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。

ステップ 4: ブローカーを削除する

Amazon MQ ブローカーを使用しない場合 (および近い将来使用することが予想される場合)、 AWS コストを削減するために Amazon MQ から削除することがベストプラクティスです。

以下の例では、 AWS Management Consoleを使用してブローカーを削除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. ブローカーリストからブローカー (などMyBroker) を選択し、「削除」を選択します。
3. Delete ***MyBroker***? ダイアログボックスで、`delete`「」と入力し、「削除」を選択します。

   ブローカーの削除には約 5 分かかります。

次のステップ

ブローカーを作成してアプリケーションを接続し、メッセージを送受信したので、次の操作を試してください。

- Creating and configuring a broker (詳細設定)

- ブローカーエンジンのバージョン、インスタンスタイプ、 CloudWatch ログ、メンテナンス設定の編集
- Creating and applying broker configurations
- Listing brokers and viewing broker details
- ActiveMQ ブローカーユーザーの作成と管理
- Rebooting a Broker
- Amazon MQ 向けの CloudWatch メトリクスへのアクセス

Amazon MQ のベストプラクティスと Amazon MQ REST API をよく理解した上で、Amazon MQ への移行を計画することもできます。

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`、`t3`) およびサイズ (`large`、`micro`) を組み合わせた説明がブローカーインスタンスタイプ (`mq.m5.large` など) になります。

トピック

- ステップ 1: RabbitMQ ブローカーを作成する
- 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
- ステップ 3: (オプション) AWS Lambda 関数に接続する
- ステップ 4: ブローカーを削除する
- 次のステップ

ステップ 1: 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。次の例は、 を使用して基本的なブローカー AWS Management Console を作成する方法を示しています。

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、「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、「高可用性対応の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 ログを含む他の 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。

   > **Note**
   >
   > 追加設定セクションには、 CloudWatch ログを有効にし、ブローカーのネットワークアクセスを設定するオプションがあります。パブリックアクセシビリティがないプライベート RabbitMQ ブローカーを作成する場合は、Virtual Private Cloud (VPC) を選択し、ブローカーにアクセスするためのセキュリティグループを設定する必要があります。

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。

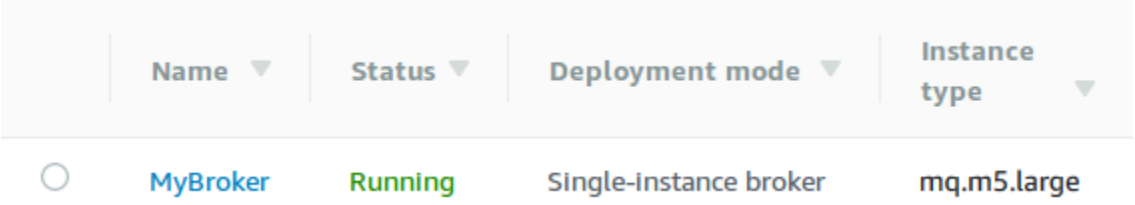
> **Important**
>
> 個人を特定できる情報 (PII) などの機密情報や秘匿性の高い情報はブローカーのユーザー名に追加しないでください。ブローカーユーザー名は、 CloudWatch ログを含む他の AWS のサービスからアクセスできます。ブローカーのユーザー名は、プライベートデータや機密データとして使用することを意図していません。

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. を選択します***MyBroker***。

   MyBroker ページの Connect セクションで、ブローカーの RabbitMQ ウェブコンソール URL をメモします。次に例を示します。

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

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.amazonaws.com:5671
```

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. ブローカーリストから、ブローカーの名前 (例:) を選択しますMyBroker。
3. ***MyBroker*** ページの「接続」セクションで、ブローカーのウェブコンソール URL とワイヤレベルの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メッセージを使用できます。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
- ログ：CreateLogGroup
- ログ：CreateLogStream
- ログ：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) と 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 Management Console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。

ステップ 4: ブローカーを削除する

Amazon MQ ブローカーを使用しない (かつ近い将来使用しないと予想される) 場合は、Amazon MQ から削除して AWS コストを削減することがベストプラクティスです。

以下の例では、 AWS Management Consoleを使用してブローカーを削除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. ブローカーリストからブローカー (例: MyBroker) を選択し、削除 を選択します。
3. 削除 ***MyBroker***？ ダイアログボックスで、`delete`「」と入力し、「削除」を選択します。

   ブローカーの削除には約 5 分かかります。

次のステップ

ブローカーを作成してアプリケーションを接続し、メッセージを送受信したので、次の操作を試してください。

- ブローカーエンジンのバージョン、インスタンスタイプ、 CloudWatch ログ、メンテナンス設定の編集
- Listing brokers and viewing broker details
- ActiveMQ ブローカーユーザーの作成と管理
- Rebooting a Broker
- Amazon MQ 向けの CloudWatch メトリクスへのアクセス

Amazon MQ のベストプラクティスと Amazon MQ REST API をよく理解した上で、Amazon MQ への移行を計画することもできます。

Amazon MQ ブローカーの管理

以下のセクションでは、Amazon MQ ブローカーの管理とメンテナンスに関する手順を説明します。

トピック

- Amazon MQ ブローカーのメンテナンス
- Amazon MQ ブローカーエンジンバージョンのアップグレード
- ブローカーステータス
- Amazon MQ ブローカーのリスト化とブローカー詳細の表示
- パブリックアクセシビリティが無効化されたブローカーウェブコンソールへのアクセス
- Amazon MQ ブローカーの再起動
- Amazon MQ ブローカーの削除
- Amazon MQ ブローカーの設定の管理
- インスタンスのタイプ
- リソースのタグ付け

Amazon MQ ブローカーのメンテナンス

Amazon MQ は、ハードウェア、オペレーティングシステム、エンジンソフトウェア、またはメッセージブローカー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。例えば、エンジンの自動マイナーバージョンアップグレードをアクティブ化した場合、またはブローカーインスタンスタイプを変更した場合、Amazon MQ は次にスケジュールされているメンテナンスウィンドウ中に変更を適用します。

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ため、複数のアベイラビリティーゾーン (AZ) にまたがる、高可用性を備えたブローカーデプロイモードを選択することをお勧めします。ブローカーのエンジンタイプに応じて、Amazon MQ は以下のマルチ AZ 配置モードを提供します。

- Amazon MQ for ActiveMQ – Amazon MQ for ActiveMQ は、高可用性のためにアクティブ/スタンバイデプロイを提供します。アクティブ/スタンバイモードでは、Amazon MQ がメンテナンス操作を一度に 1 インスタンスずつ実行して、少なくとも 1 つのインスタンスが利用可能であることを保証します。さらに、メンテナンスウィンドウが 1 週間のさまざまな時点に分散されたブローカーのネットワークを設定することもできます。

- Amazon MQ for RabbitMQ – Amazon MQ for RabbitMQ は、高可用性のためにクラスターデプロイを提供します。クラスターデプロイでは、Amazon MQ が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 つのノードが常に実行され続けるようにします。

メンテナンスウィンドウ中、またはその後もブローカーが効率的に動作することを確実にするための Amazon MQ の推奨ベストプラクティスに関する詳細については、ブローカーのエンジンタイプに対応する以下のドキュメントを参照してください。

- the section called “Amazon MQ for ActiveMQ のベストプラクティス”
- the section called “Amazon MQ for RabbitMQ のベストプラクティス”

メンテナンスは週に 1 回、指定された時刻に実行されるようにスケジュールでき、最長で 2 時間かかります。これは、Amazon MQ からのメンテナンスアクションがスケジュールされ、開始される時間帯を設定します。

メンテナンスウィンドウは、ブローカーを初めて作成するときにスケジュールする、またはブローカー設定を更新することによってスケジュールすることができます。次のトピックでは、 AWS Management Console、 AWS CLI、および 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調整について説明します。

トピック

- ブローカーメンテナンスウィンドウの調整

ブローカーメンテナンスウィンドウの調整

選択したメンテナンスウィンドウ中に、Amazon MQ は自動マイナーバージョンアップグレードなどの保留中の変更を実行します。ブローカーのメンテナンスウィンドウを調整するには、 AWS Management Console、 AWS CLI、または Amazon MQ API を使用できます。

> **Important**
>
> 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。Amazon MQ は、重要なソフトウェアパッチとセキュリティパッチ、および重要なハードウェアアップグレードが無期限に延期されることがないように、メンテナンスウィンドウの調整を 4 回に制限します。
> ブローカーメンテナンスウィンドウが完了すると、Amazon MQ がこの制限をリセットするため、次回のメンテナンスウィンドウまでスケジュールを調整できるようになります。

ブローカーのメンテナンスウィンドウを調整しても、ブローカーの可用性には影響しません。

AWS Management Console

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には AWS Management Console

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. ページの最下部までスクロールし、[Save] (保存) をクリックします。メンテナンスウィンドウは直ちに調整されます。
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

を使用してブローカーメンテナンスウィンドウを調整するには 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。

- --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– MONDAY| TUESDAY | WEDNESDAY | THURSDAY | FRIDAY | SATURDAY | SUNDAY の構文で指定する曜日です。
  - 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

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確認します。 broker-idはパスパラメータとして指定されます。

```
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ます。

Amazon MQ は、セマンティックバージョニング仕様に従ってバージョン番号を として整理しますX.Y.Z。Amazon MQ 実装では、 X はメジャーバージョン、 Yはマイナーバージョン、 Zはパッチバージョン番号を表します。アップグレードには以下の 2 つのタイプがあります。

- 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、バージョン 1.0 からバージョン 2.0 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アップグレード — マイナーエンジンまたはパッチエンジンのバージョン番号のみが変更されたときに発生します。例えば、バージョン 1.5 からバージョン 1.6 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。

特定のブローカーエンジンタイプごとのメジャーバージョン管理とマイナーバージョン管理の詳細については、以下のトピックを参照してください。

- the section called “バージョン管理”

- the section called “バージョン管理”

ブローカーは、いつでも、次にサポートされているメジャー、マイナー、またはパッチ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中にブローカーをサポートされている最新のパッチバージョンにアップグレードします。 Amazon MQ 自動マイナーバージョンアップグレードを有効にしない場合、現在のマイナーバージョンがサポート終了になると、Amazon MQ は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。

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。

以下のトピックでは、ブローカーエンジンバージョンを手動でアップグレードする方法と、自動マイナーバージョンアップグレードをアクティブにする方法について説明します。

トピック

- エンジンバージョンの手動アップグレード
- マイナーエンジンバージョンの自動アップグレード
- エンジンバージョンのサポート終了カレンダー

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を新しい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するには、、 AWS CLI、または Amazon MQ API AWS Management Consoleを使用できます。

AWS Management Console

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 Management Console

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. [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。

5. 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。
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、ブローカーはオフラインになります。

7. [Apply] (適用) をクリックして、変更の適用を終了します。

AWS CLI

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、ブローカーはオフラインになります。

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、ブローカーはオフラインになります。

マイナーエンジンバージョンの自動アップグレード

ブローカーの自動マイナーバージョンアップグレードを、初めてブローカーを作成するときにアクティブにするか、ブローカー設定を変更することによってアクティブにするかは、ユーザーが制御できます。既存のブローカーの自動マイナーバージョンアップグレードを有効にするには、、 AWS CLI、または Amazon MQ API AWS Management Consoleを使用できます。

AWS Management Console

を使用して自動マイナーバージョンアップグレードを有効にするには AWS Management Console

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、[Enable automatic minor version upgrades](自動マイナーバージョンアップグレードの有効化) を選択します。

   > **Note**
   >
   > このオプションが既に選択されている場合は、何も変更する必要はありません。

5. ページの最下部で [Save] (保存) をクリックします。

AWS CLI

経由で自動マイナーバージョンアップグレードを有効にするには AWS CLI、update-broker CLI コマンドを使用して、次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。
- `--auto-minor-version-upgrade` – 自動マイナーバージョンアップグレードオプションをアクティブにします。

```
aws mq update-broker --broker-id broker-id --auto-minor-version-upgrade
```

ブローカーの自動マイナーバージョンアップグレードを非アクティブにする場合は、`--no-auto-minor-version-upgrade` パラメータを使用します。

Amazon MQ API

Amazon MQ API を使用して自動マイナーバージョンアップグレードを有効にするには、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 `autoMinorVersionUpgrade` プロパティを使用して、自動マイナーバージョンアップグレードをアクティブにします。

```
{
    "autoMinorVersionUpgrade": "true"
}
```

ブローカーの自動アマイナーバージョンップグレードを非アクティブにする場合は、リクエストペイロードで `"autoMinorVersionUpgrade": "false"` を設定します。

エンジンバージョンのサポート終了カレンダー

Amazon MQ バージョンのサポート終了カレンダーは、ブローカーエンジンのバージョンがサポート終了になると通知します。エンジンバージョンのサポートが終了すると、Amazon MQ はこのバージョンのすべてのブローカーを次に利用可能な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を終了する少なくとも 90 日前に通知します。

バージョンサポートカレンダーを表示するには、Amazon MQ ??? for ActiveMQ」とAmazon MQ ??? for RabbitMQ」を参照してください。

ブローカーステータス

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、the section called “トラブルシューティン

コンソール	API	説明
		グ:Amazon MQ のアクションに必要なコード" のリストからアクション必須コードを選択します。

Amazon MQ ブローカーのリスト化とブローカー詳細の表示

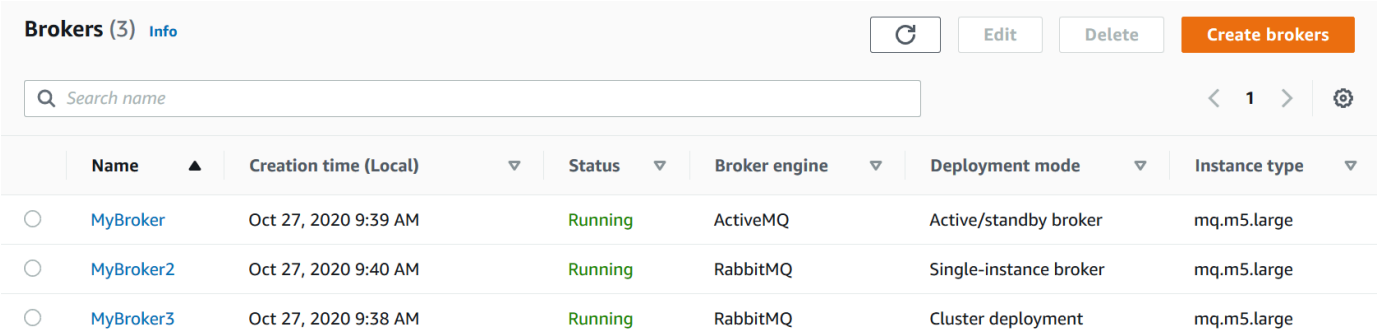
Amazon MQ によるブローカーの作成をリクエストするときは、作成プロセスに約 15 分かかる場合があります。

次の例では、AWS Management Console を使用して、現在のリージョンにあるすべてのブローカーをリストし、ブローカーが存在することを確認する方法を示しています。

ブローカーをリストしてブローカーの詳細を表示する

1. Amazon MQ コンソールにサインインします。

   現在のリージョン内にあるブローカーが表示されます。

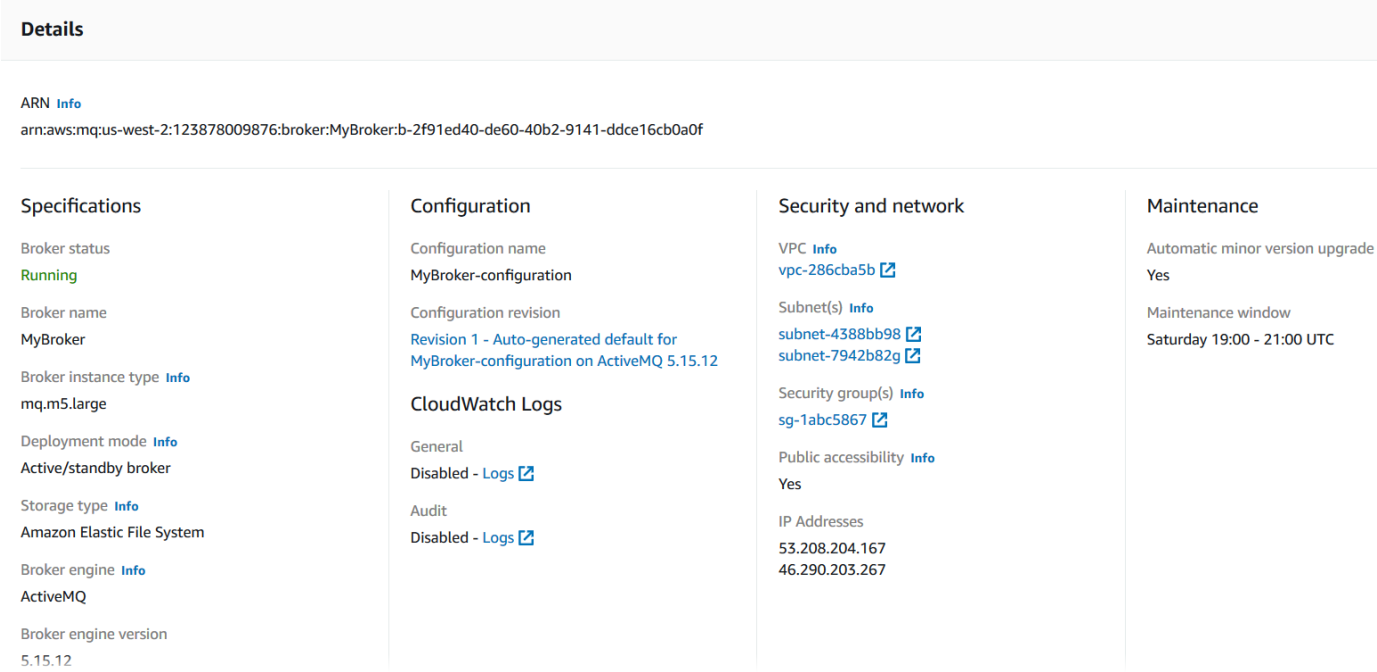


   各ブローカーについて、以下の情報が表示されます。

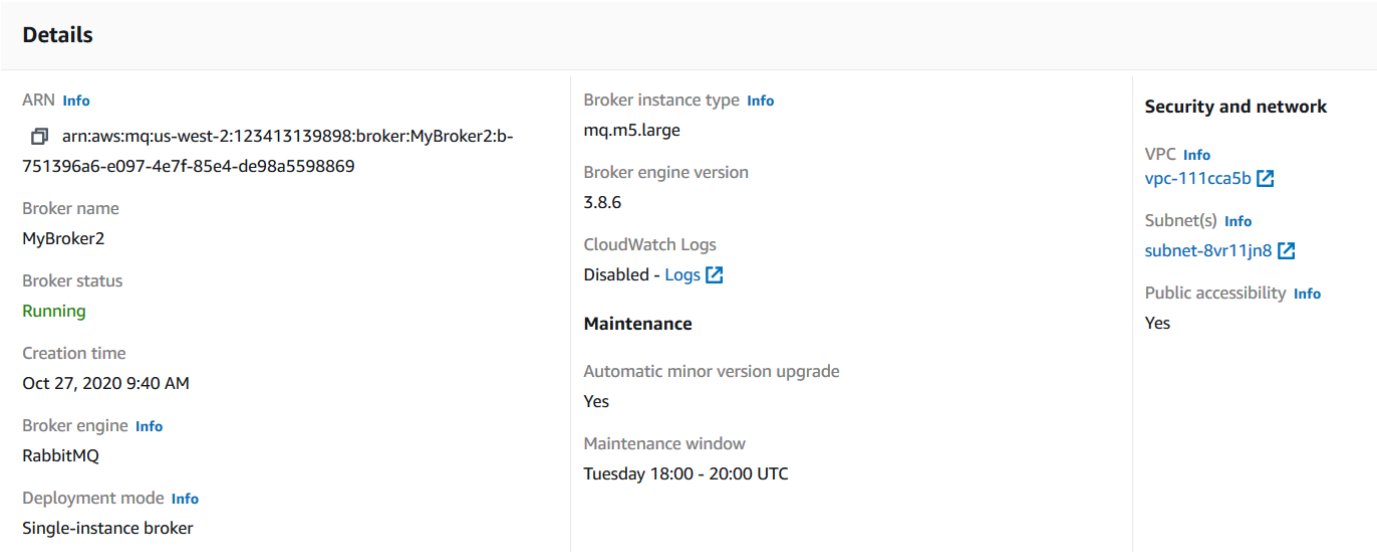
   - 名前
   - 作成日
   - ステータス
   - [デプロイモード]
   - インスタンスタイプ

2. ブローカーの名前を選択します。

ActiveMQ ブローカーについては、[***MyBroker***] (ブローカー) ページに、ブローカーに関する設定済みの [Details] (詳細) が表示されます。

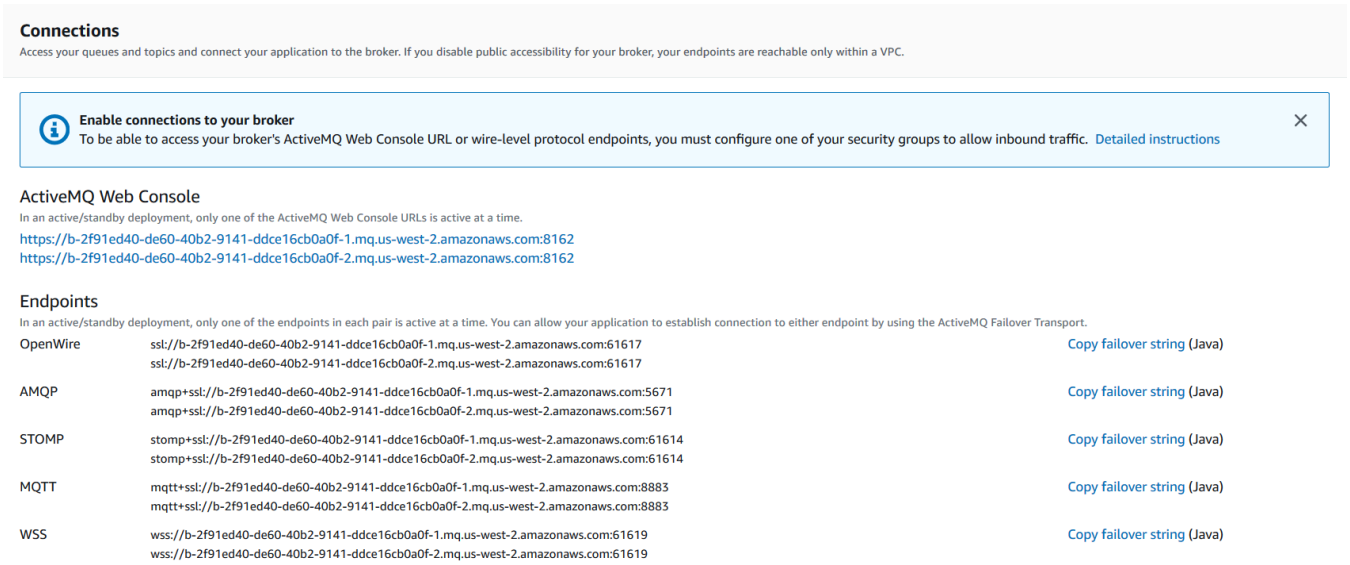


Amazon MQ for RabbitMQ ブローカーについては、以下にあるように、***MyBroker2*** ページの [Details] (詳細) セクションで、選択した設定を確認できます。

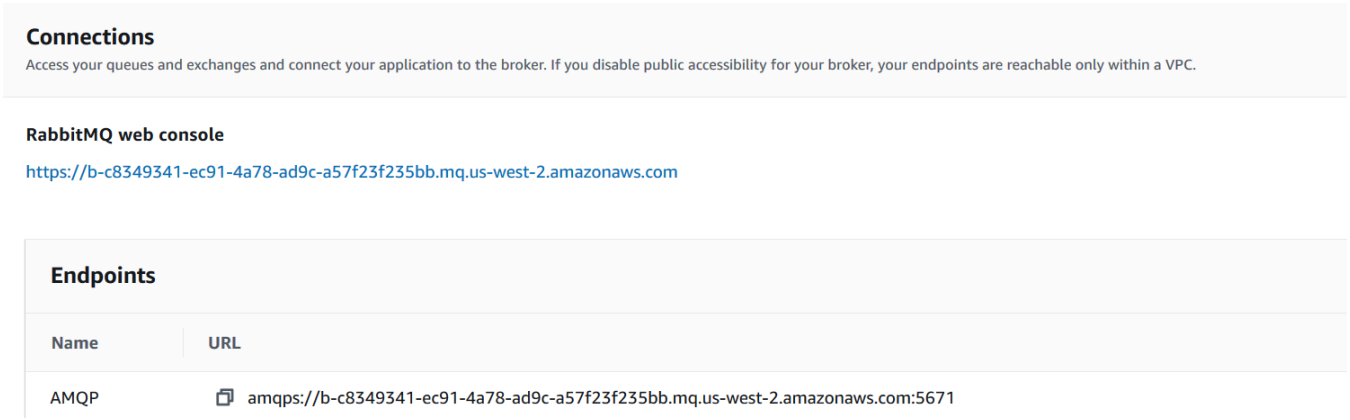


[Details] (詳細) セクションに、以下の情報が表示されます。

- Amazon MQ for ActiveMQ ブローカーについては、[Connections] (接続) セクションにウェブコンソール URL とワイヤレベルプロトコルのエンドポイントが表示されます。



Amazon MQ for RabbitMQ ブローカーについては、[Connections] (接続) セクションにウェブコンソール URL とセキュア AMQP エンドポイントが表示されます。



- Amazon MQ for ActiveMQ ブローカー の場合は、ブローカーに関連付けられたユーザーが [Users] (ユーザー) セクションに表示されます。

> **Important**
>
> Amazon MQ for RabbitMQ ブローカーでは、AWS Management Console および Amazon MQ API を使用したユーザーの管理がサポートされていません。

パブリックアクセシビリティが無効化されたブローカーウェブコンソールへのアクセス

ブローカーに対するパブリックアクセシビリティを無効にしている場合にブローカーのウェブコンソールにアクセスするには、以下のステップを実行する必要があります。

> **Note**
>
> VPC とセキュリティグループの名前は、次の例に固有のものです。

前提条件

以下のステップを実行するには、次の設定を行う必要があります。

- VPC
  - Amazon MQ ブローカーがアタッチされている VPC で、インターネットゲートウェイがない `private-vpc` という名前の VPC。
  - インターネットゲートウェイがある、`public-vpc` という名前の 2 つ目の VPC。
  - パブリック VPC 内の EC2 インスタンスがプライベート VPC 内の Amazon EC2 インスタンスと通信できるように、VPC ピアリングの使用などで、両方の VPC が接続されている必要があります。
  - VPC ピアリングを使用する場合は、両 VPC のルートテーブルを ピア接続用に設定する必要があります。
- セキュリティグループ
  - Amazon MQ ブローカーを作成するために使用された、`private-sg` という名前のセキュリティグループ。
  - `public-vpc` VPC の EC2 インスタンスで使用する第 2 のセキュリティグループ。名前は `public-sg`。
  - `private-sg` を使用して、`public-sg` からのインバウンド接続を許可します。このセキュリティグループを ActiveMQ の場合はポート 8162、RabbitMQ の場合はポート 443 に制限することをお勧めします。
  - `public-sg` を使用して、お使いのマシンからのインバウンド接続をポート 22 で許可します。

パブリックアクセシビリティが無効化されているブローカーのウェブコンソールにアクセスする

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。

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。

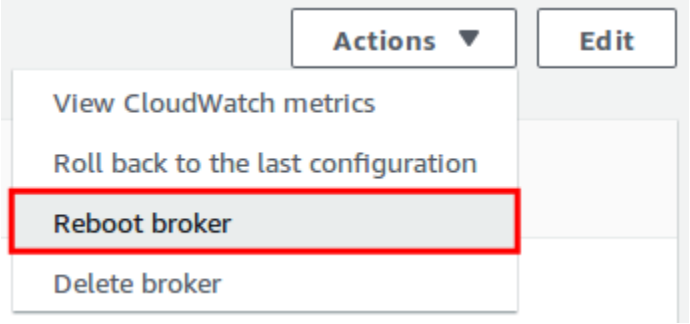
> **Note**
>
> ActiveMQ ブローカーが応答しない場合、ブローカーを再起動して障害状態から復旧できます。

以下の例では、AWS Management Console 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 (かつ近い将来使用することがないと思われる) 場合は、ブローカーを Amazon MQ から削除して AWS 料金を削減することがベストプラクティスです。

以下の例では、AWS Management Console 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 ブローカーの設定の管理

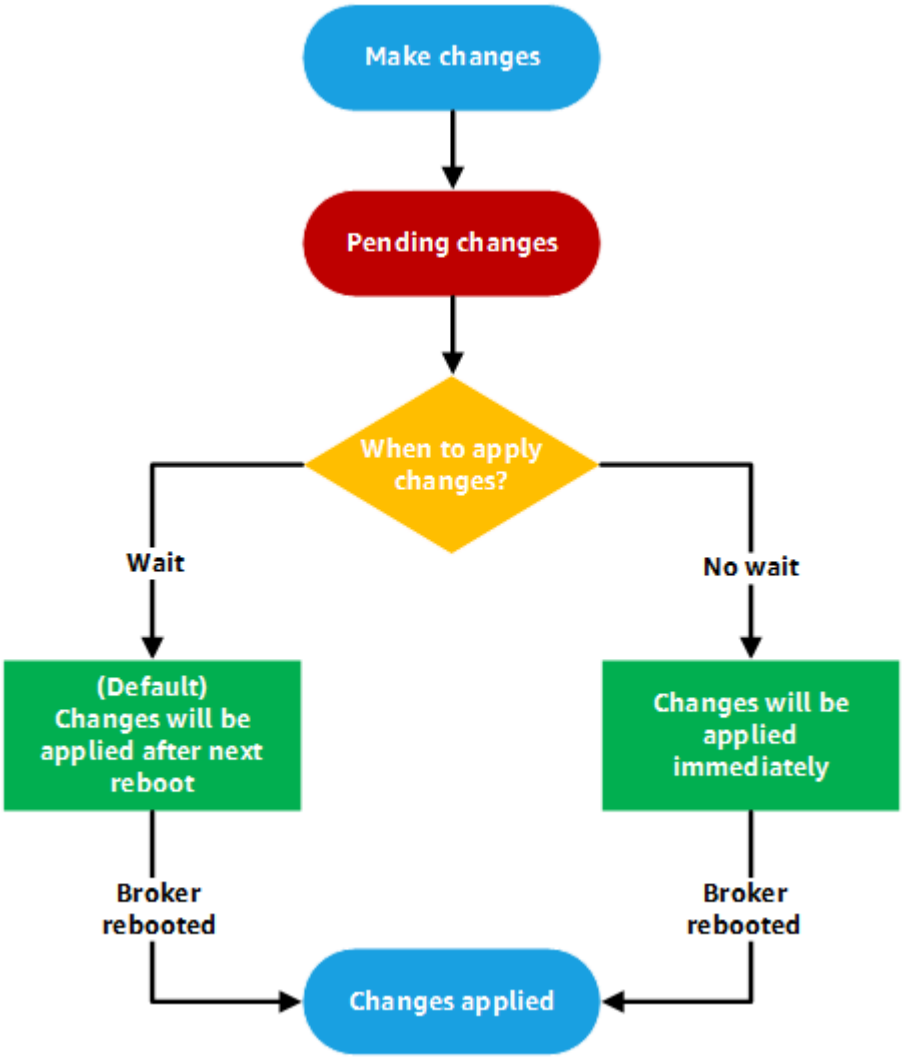
設定には、ブローカーのすべての設定が含まれています。設定は、ブローカーを作成する前に作成することができます。次に、設定を 1 つ以上のブローカーに適用できます。

Amazon MQ ブローカー設定のライフサイクル

設定リビジョン、または ActiveMQ ユーザーを変更しても、変更は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。詳細については、「Amazon MQ ブローカー設定のライフサイクル」を参照してください。

次の図は設定のライフサイクルを示しています。

> **Important**
>
> 次にスケジュールされたメンテナンスウィンドウが再起動をトリガーします。次にスケジュールされたメンテナンスウィンドウの前にブローカーが再起動された場合、変更は再起動後に適用されます。



ActiveMQ の場合、設定には、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)。ActiveMQ ブローカー設定の作成、適用、編集の詳細については、「Creating and applying broker configurations」を参照してください。

RabbitMQ の場合、設定には、ブローカーのすべての設定が Cuttlefish 形式で含まれています。RabbitMQ ブローカー設定の作成、適用、編集の詳細については、「Creating and applying broker configurations」を参照してください。

インスタンスのタイプ

ブローカーインスタンスのクラス (`m5`、`t3`) およびサイズ (`large`、`micro`) を組み合わせた説明がブローカーインスタンスタイプ (`mq.m5.large` など) になります。以下の表には、サポートされている各エンジンタイプに利用できる Amazon MQ ブローカーインスタンスタイプがリストされています。

トピック

- Amazon MQ for ActiveMQ インスタンスタイプ
- Amazon MQ for RabbitMQ インスタンスタイプ

Amazon MQ for ActiveMQ インスタンスタイプ

> **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。詳細については、「Storage」を参照してください。

インスタンスタイプ	vCPU	メモリ (GiB)	ネットワークパフォーマンス	推奨使用法
mq.t2.micro	1	1	低	評価
mq.t3.micro	2	1	低	評価
mq.m4.large	2	8	中	本番稼働
mq.m5.large	2	8	高い	本番稼働
mq.m5.xlarge	4	16	高い	本番稼働
mq.m5.2xlarge	8	32	高い	
mq.m5.4xlarge	16	64	高い	

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 RabbitMQ インスタンスタイプ

> **Important**
>
> 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。

インスタンスタイプ	vCPU	メモリ (GiB)	ネットワークパフォーマンス	ユースケース
mq.t3.micro	2	1	低	評価 **Important** mq.t3.micro インスタンスタイプはクラスターデプロイをサポートしません。
mq.m5.large	2	8	高い	本番稼働
mq.m5.xlarge	4	16	高い	本番稼働
mq.m5.2xlarge	8	32	高い	
mq.m5.4xlarge	16	64	高い	

リソースのタグ付け

Amazon MQ は、コスト配分を追跡するために役立つリソースのタグ付けをサポートします。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。

トピック

- コスト割り当てのタグ付け
- Amazon MQ コンソールでのタグの管理
- Amazon MQ API アクションを使用した管理

コスト割り当てのタグ付け

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、自分のコスト構造を反映できます。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管理

新しいリソースにタグを追加する

Amazon MQ では、リソースの作成時にタグを追加することができます。Amazon MQ コンソールで作成しているリソースにすばやくタグを追加できます。

新しいブローカーを作成するときにタグを追加するには。

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。

既存のリソースのタグの表示と管理

Amazon MQ では、Amazon MQ コンソールでリソースのタグを表示し、管理することができます。そのリソースの詳細ページでタグを編集することで、個々のリソースのタグを管理できます。Amazon MQ リソースのタグを編集する

1. Amazon MQ コンソールで、[Brokers] (ブローカー) または [Configurations] (設定) をクリックします。

   [タグ] セクションで、そのリソースの既存のタグを確認します。
2. 新規または既存のタグ管理を追加するには、[編集] (または既存のタグがない場合は [タグの作成]) を選択します。
3. リソースのタグを更新します。

   - 既存のタグを変更するには、[キー] と [値] フィールドを編集します。
   - 既存のタグを削除するには、[削除] を選択します。
   - 新しいタグを追加するには、[タグの追加] を選択し、[キー] と [値] を入力します。
4. [Save] (保存) をクリックします。

Amazon MQ API アクションを使用した管理

Amazon MQ では、REST API を使用してリソースのタグを表示し、管理することができます。

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 Management Console、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。

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。

トピック

- ActiveMQ エンジン
- ActiveMQ チュートリアル
- Amazon MQ for ActiveMQ のベストプラクティス
- Amazon MQ for ActiveMQ のクロスリージョンデータレプリケーション
- Amazon MQ for ActiveMQ のクォータ

ActiveMQ エンジン

このセクションでは、ActiveMQ ブローカーの基本的要素、ActiveMQ ブローカーのアーキテクチャオプションの概要、およびブローカーの設定パラメータについて説明し、Java Message Service (JMS) を使用した実用例を提供します。

トピック

- 基本的要素
- ブローカーのアーキテクチャ
- Amazon MQ for ActiveMQ ブローカーの設定
- Amazon MQ for ActiveMQ エンジンバージョンの管理
- ActiveMQ での Java Message Service (JMS) の使用の実用例

基本的要素

このセクションでは、ActiveMQ on Amazon MQ を理解するうえで不可欠な主要概念を説明します。

トピック

- ブローカー
- ブローカーインスタンスタイプ
- 構成
- ユーザー
- Storage

ブローカー

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`、`t3`) およびサイズ (`large`、`micro`) を組み合わせた説明がブローカーインスタンスタイプ (`mq.m5.large` など) になります。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、「Broker Architecture」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。

- Creating and configuring a broker
- 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 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

ブローカーインスタンスタイプ

> **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。詳細については、「Storage」を参照してください。

インスタンスタイプ	vCPU	メモリ (GiB)	ネットワークパフォーマンス	メモ
mq.t2.micro	1	1	低	Amazon MQ の基本的な評価には mq.t2.micro インスタンスタイプを使用します。このインスタンスタイプ (単一インスタンスブローカーのみ) は AWS 無料利用枠対象です。 **Note** 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

インスタンスタイプ	vCPU	メモリ (GiB)	ネットワークパフォーマンス	メモ
				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。

インスタンスタイプ	vCPU	メモリ (GiB)	ネットワークパフォーマンス	メモ
mq.t3.micro	2	1	低	Amazon MQ の基本的な評価には mq.t3.micro インスタンスタイプを使用します。このインスタンスタイプ (単一インスタンスブローカーのみ) は、AWS 無料利用枠の対象となります。
mq.m4.large	2	8	中	既存のブローカーのデプロイとの互換性のある mq.m4.large インスタンスタイプを使用します。新しいブローカーには mq.m5.* インスタンスの使用が推奨されます。
mq.m5.large	2	8	高い	一般的な開発、テスト、および本稼動のワークロードには、mq.m5.large インスタンスを使用します。

インスタンスタイプ	vCPU	メモリ (GiB)	ネットワークパフォーマンス	メモ
mq.m5.xlarge	4	16	高い	高いスループットを必要とする一般的な開発、テスト、および本稼働のワークロードには、mq.m5.xlarge 、mq.m5.2xlarge および mq.m5.4xlarge インスタンスタイプを使用します。 **Note** システムが永続的なメッセージを使用する場合、そのスループットはメッセージが消費される速度に依存します。メッセージがすぐに消費されない場合、永続
mq.m5.2xlarge	8	32	高い	
mq.m5.4xlarge	16	64	高い	

インスタンスタイプ	vCPU	メモリ (GiB)	ネットワークパフォーマンス	メモ
				的なメッセージのあるより大きなインスタンスタイプを使用することは、システムのスループットを向上させない場合があります。この場合には、concurrentStoreAndDispatchQueues に false 属性を設定することが推奨されます。詳細については、「低速コンシューマーのキュー

インスタンスタイプ	vCPU	メモリ (GiB)	ネットワークパフォーマンス	メモ
				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。

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。

構成

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。

> **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。詳細については、「Amazon MQ ブローカー設定のライフサイクル」を参照してください。現在、設定を削除することはできません。

設定を作成、編集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。

- Creating and applying broker configurations
- Configurations
- Amazon MQ Broker Configuration Parameters

設定に対して行った変更を追跡するために、設定リビジョンを作成できます。詳細については、「Creating and applying broker configurations」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

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。詳細については、「Amazon MQ ブローカー設定のライフサイクル」を参照してください。

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。

- 認証
- 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と管理
- Users

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

Storage

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	Persistence	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
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。

ブローカーのアーキテクチャ

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。

> **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

トピック

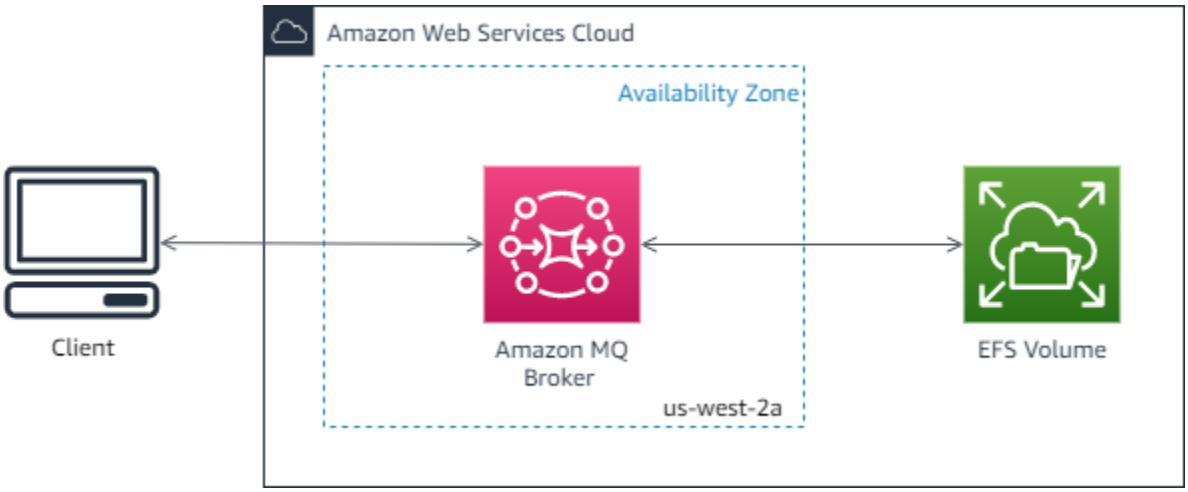
- Amazon MQ 単一インスタンスブローカー
- 高可用性対応の Amazon MQ アクティブ/スタンバイブローカー
- ブローカーの Amazon MQ ネットワーク

Amazon MQ 単一インスタンスブローカー

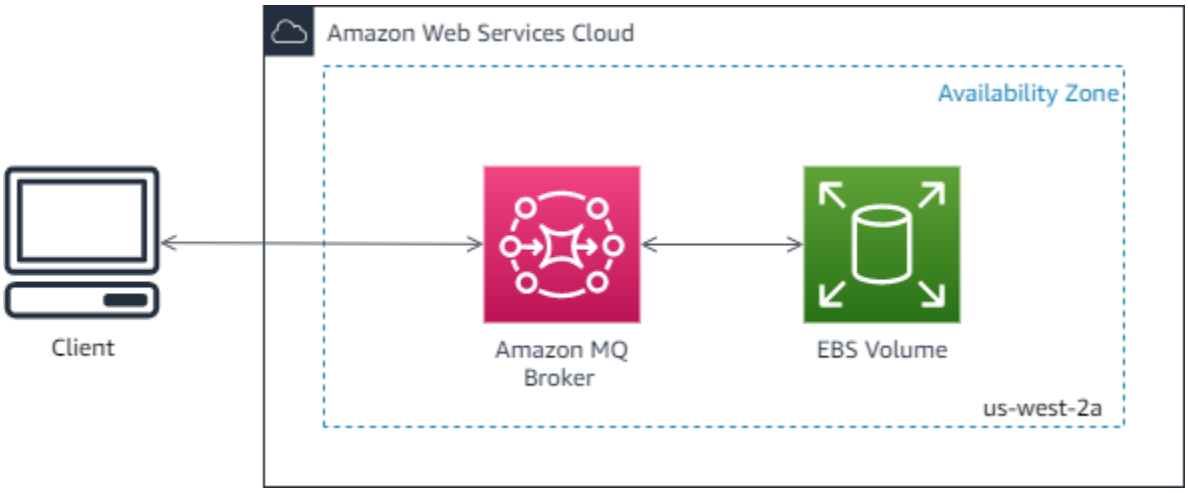
単一インスタンスブローカーは、1 つのアベイラビリティーゾーン内の 1 つのブローカーで構成されます。ブローカーは、アプリケーション、および Amazon EBS または Amazon EFS ストレージボリュームと通信します。Amazon EFS ストレージボリュームは、複数のアベイラビリティーゾーン (AZ) にまたがってデータを冗長的に保存することにより、最高レベルの耐久性と可用性を実現する

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。



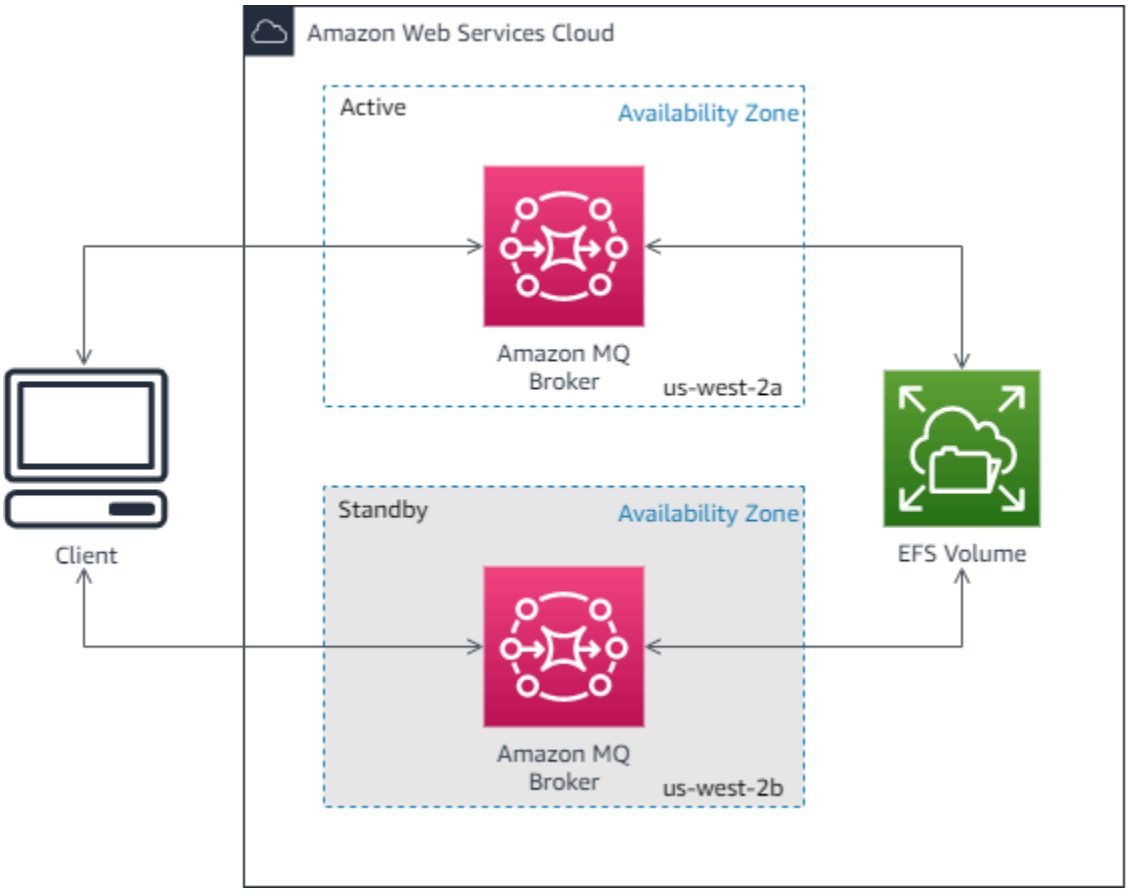
高可用性対応の Amazon MQ アクティブ/スタンバイブローカー

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。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できます。

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。



ブローカーの Amazon MQ 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は、高可用性やスケーラビリティなどのアプリケーションのニーズに応じて、さまざまなトポロジ (コンセントレータ、ハブアンドスポーク、ツリー、またはメッシュなど) で設定できます。例えば、ブローカーのハブアンドスポークネットワークは耐障害性を高めることができ、1 つのブローカーが到達不能な場合にはメッセージを保存します。コンセントレータトポロジを使用するブローカーのネットワークは、多数の着信メッセージの負荷をより良く処理するために、着信メッセージを受け入れる多数のブローカーからメッセージを収集し、それらをより中核的なブローカーに集中させます。

チュートリアル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。

- 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`
- `######ConnectionStart###`
- ActiveMQ ドキュメントの「ブローカーのネットワーク」

ブローカーのネットワークを使用すると、以下のような利点があります。

- ブローカーのネットワークを作成すると、ブローカーインスタンスを追加することで、総スループットと最大プロデューサーおよびコンシューマー接続数を増やすことができます。
- プロデューサーとコンシューマーが複数のアクティブなブローカーインスタンスを認識できるようにすることで、可用性を向上させることができます。これにより、現在接続しているインスタンスが使用できなくなった場合に、新しいインスタンスに再接続できます。
- プロデューサーとコンシューマーはブローカーネットワーク内の別のノードにすぐに再接続できるため、また、スタンバイブローカーインスタンスが昇格するのを待つ必要がないため、ブローカーのネットワーク内でのクライアントの再接続は、高可用性のためのアクティブ/スタンバイブローカーよりも高速です。

トピック

- 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どのように認証情報を処理しますか?

- サンプル設計図
- ブローカーのネットワークのトポロジ
- クロスリージョン
- 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

ブローカーのネットワークの仕組み

Amazon MQ は、ActiveMQ のブローカーのネットワーク機能をさまざまな方法でサポートします。まず、ネイティブ ActiveMQ と同じように、各ブローカーの設定内のパラメータを編集してブローカーのネットワークを作成できます。次に、Amazon MQ には、AWS CloudFormation を使用してブローカーのネットワークの作成を自動化するサンプル設計図があります。Amazon MQ コンソールからこれらのサンプル設計図を直接デプロイする、または関連する AWS CloudFormation テンプレートを編集して独自のトポロジと設定を作成することが可能です。

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ブローカー同士を接続することによって確立されます。接続されると、これらのブローカーはメッセージ転送を提供します。例えば、Broker1 が Broker2 へのネットワークコネクターを確立する場合、ブローカーにキューまたはトピックのコンシューマーが存在すると、Broker1 のメッセージは Broker2 に転送されます。ネットワークコネクターが `duplex` 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。ネットワークコネクターはブローカーの設定で設定されます。「構成」を参照してください。例えば、ブローカー設定の `networkConnector` エントリの例は以下のようになります。

```
<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>
```

ブローカーのネットワークは、メッセージがあるブローカーインスタンスから別のブローカーインスタンスに確実に流れ、対応するコンシューマーを持つブローカーインスタンスにのみメッセージを転送するようにします。ネットワーク内でブローカーインスタンスが互いに隣接しているという利点のために、ActiveMQ は、ネットワークに接続したり切断したりするプロデューサーとコンシューマーに関するメッセージをアドバイザリトピックに送信します。ブローカーインスタンスが特定の送信先から消費するコンシューマーに関する情報を受信すると、ブローカーインスタンスはメッセージの転送を開始します。詳細については、ActiveMQ ドキュメントの「アドバイザリトピック」を参照してください。

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。

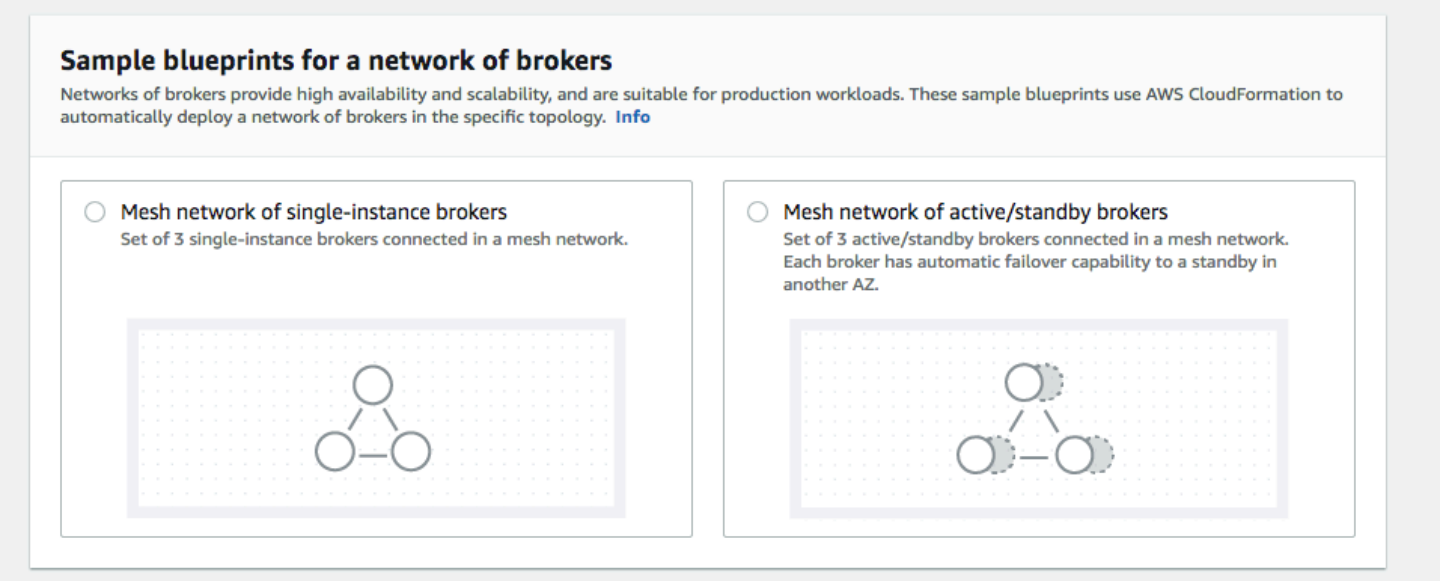
> **Important**
>
> <networkConnector> には password 属性を指定しないでください。パスワードが Amazon MQ コンソールに表示されてしまうため、プレーンテキストのパスワードをブローカー設定ファイルに保存することは推奨されません。詳細については、「Configure Network Connectors for Your Broker」を参照してください。

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。詳細については、Creating and Configuring a Network of Brokersチュートリアルの「前提条件」を参照してください。

サンプル設計図

Amazon MQ では、ブローカーのネットワークの使用を開始するためのサンプル設計図が提供されています。これらのサンプルの設計図は、ブローカーのネットワーク展開、および AWS CloudFormation を使用したすべての関連リソースを作成します。利用可能な 2 つのサンプル設計図は次のとおりです。

1. シングルインスタンスブローカーのメッシュネットワーク
2. アクティブ/スタンバイブローカーのメッシュネットワーク



[ブローカーの作成] ページから、サンプルの設計図を 1 つ選択し、[次へ] を選択します。リソースが作成されたら、Amazon MQ コンソールで生成されたブローカーとその設定を確認します。

ブローカーを作成し、ブローカー設定でさまざまな `networkConnector` 要素を設定することで、さまざまなトポロジでブローカーのネットワークを作成できます。ブローカーのネットワーク設定の詳細については、ActiveMQ ドキュメントの「ブローカーのネットワーク」を参照してください。

ブローカーのネットワークのトポロジ

ブローカーをデプロイし、その設定で `networkConnector` エントリを設定することで、さまざまなネットワークトポロジを使用してブローカーのネットワークを構築できます。ネットワークコネクターは、接続されているブローカー間でオンデマンドメッセージ転送を提供します。接続は、メッセージがブローカー間で双方向に転送される二重、または転送が一方のブローカーから他方のブローカーにのみ伝達される二重ではないように構成できます。たとえば、Broker1 と Broker2 の間に二重接続があり、コンシューマーが存在する場合は、メッセージはお互いに転送されます。

二重ネットワークコネクターでは、メッセージは各ブローカーから他のブローカーに転送されます。これらはオンデマンドで転送されます。Broker1 のメッセージに対するコンシューマーが Broker2 に

ある場合は、メッセージが転送されます。同様に、Broker2 のメッセージに対するコンシューマーが Broker1 にある場合も、メッセージが転送されます。

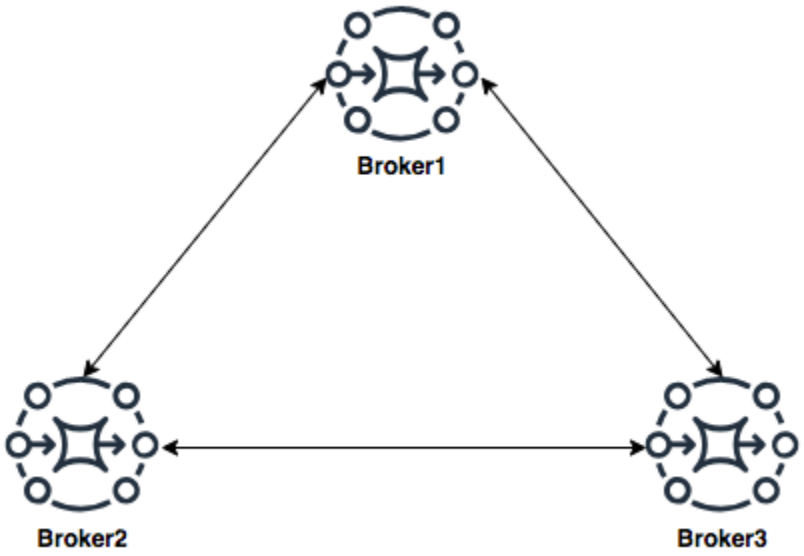
非二重接続の場合、メッセージは一方のブローカーから他方のブローカーにのみ転送されます。この例では、Broker1 のメッセージに対するコンシューマーが Broker2 にある場合に、メッセージが転送されます。しかし、メッセージは Broker2 から Broker1 に転送されません。

二重と非二重の両方のネットワークコネクターを使用して、任意の数のネットワークトポロジでブローカーのネットワークを構築できます。

> **Note**
>
> 各ネットワークトポロジの例で、`networkConnector` 要素は、接続先ブローカーのエンドポイントを参照しています。`uri` 属性のブローカーエンドポイントエントリをブローカーのエンドポイントに置き換えます。「Listing brokers and viewing broker details」を参照してください。

メッシュトポロジ

メッシュトポロジは、すべて互いに接続されている複数のブローカーを提供します。この簡単な例では、3 つのシングルインスタンスブローカーを接続していますが、より多くのブローカーをメッシュとして設定できます。



このトポロジ、およびブローカーのアクティブ/スタンバイペアのメッシュが含まれるトポロジは、Amazon MQ コンソールのサンプル設計図を使用して作成できます。これらのサンプル設計図デプロイを作成して、機能しているブローカーのネットワークを確認し、それらがどのように設定されているかを確認できます。

Broker2 と Broker3 の両方に二重接続を確立し、Broker2 と Broker3 の間に単一の二重接続を確立するネットワークコネクターを Broker1 に追加することで、このように 3 ブローカーメッシュネットワークを設定できます。

Broker1 のネットワークコネクター:

```
<networkConnectors>
    <networkConnector name="connector_1_to_2" userName="myCommonUser" duplex="true"
        uri="static:(ssl://b-9876l5k4-32ji-109h-8gfe-7d65c4b132a1-2.mq.us-
east-2.amazonaws.com:61617)"/>
    <networkConnector name="connector_1_to_3" userName="myCommonUser" duplex="true"
        uri="static:(ssl://b-743c885d-2244-4c95-af67-a85017ff234e-3.mq.us-
east-2.amazonaws.com:61617)"/>
</networkConnectors>
```

Broker2 のネットワークコネクター:

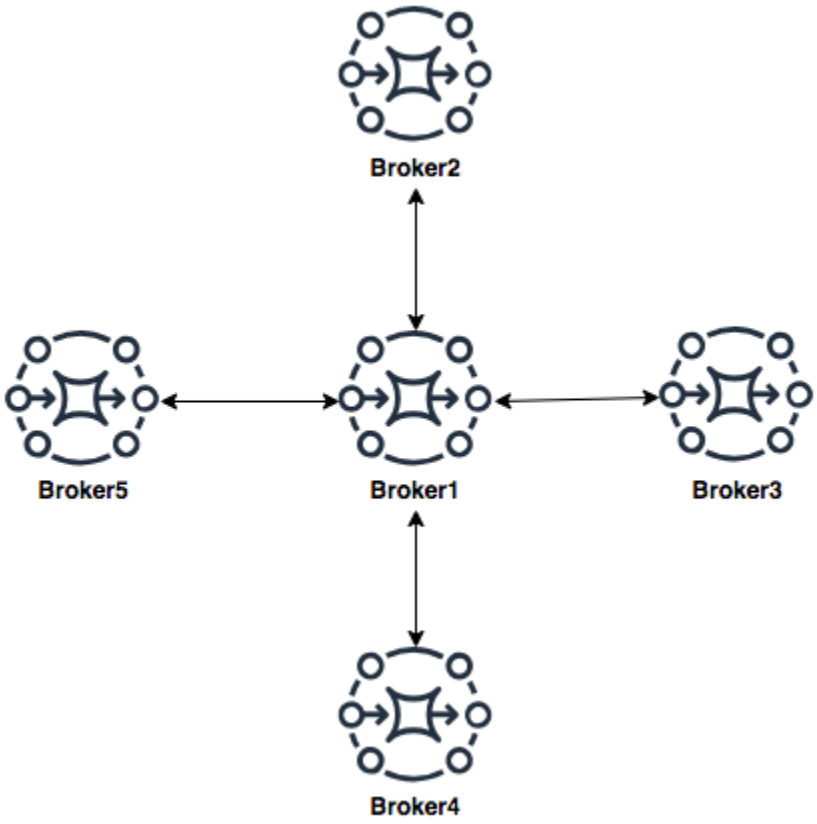
```
<networkConnectors>
    <networkConnector name="connector_2_to_3" userName="myCommonUser" duplex="true"
```

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

Broker1 と Broker2 の設定に上記のコネクターを追加することで、これら 3 つのブローカー間に、オンデマンドですべてのブローカー間でメッセージを転送するメッシュを作成できます。詳細については、「Amazon MQ Broker Configuration Parameters」を参照してください。

ハブアンドスポークトポロジ

ハブアンドスポークトポロジでは、スポークのブローカーが中断してもメッセージは保存されます。メッセージは一斉に転送され、中心的な Broker1 だけがネットワークのオペレーションに不可欠です。



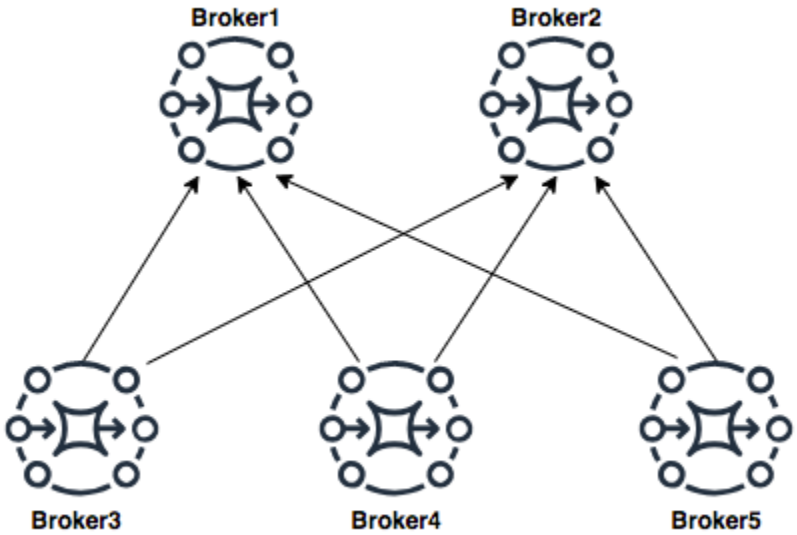
この例にあるブローカーのハブアンドスポークネットワークを設定するには、Broker1 の設定にあるスポーク上の各ブローカーに `networkConnector` を追加できます。

```
<networkConnectors>
```

```
    <networkConnector name="connector_hub_and_spoke_2" userName="myCommonUser"
 duplex="true"
        uri="static:(ssl://b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-
east-2.amazonaws.com:61617)"/>
    <networkConnector name="connector_hub_and_spoke_3" userName="myCommonUser"
 duplex="true"
        uri="static:(ssl://b-9876l5k4-32ji-109h-8gfe-7d65c4b132a1-2.mq.us-
east-2.amazonaws.com:61617)"/>
    <networkConnector name="connector_hub_and_spoke_4" userName="myCommonUser"
 duplex="true"
        uri="static:(ssl://b-743c885d-2244-4c95-af67-a85017ff234e-3.mq.us-
east-2.amazonaws.com:61617)"/>
    <networkConnector name="connector_hub_and_spoke_5" userName="myCommonUser"
 duplex="true"
        uri="static:(ssl://b-62a7fb31-d51c-466a-a873-905cd660b553-4.mq.us-
east-2.amazonaws.com:61617)"/>
</networkConnectors>
```

コンセントレータトポロジ

このトポロジ例では、一番下の 3 つのブローカーが多数の接続を処理でき、それらのメッセージは Broker1 と Broker2 に集中しています。他の各ブローカーは、より中央のブローカーへの非二重接続を持っています。このトポロジの容量を拡張するために、メッセージを受信してそれらのメッセージを Broker1 と Broker2 に集中させる追加のブローカーを追加することができます。



このトポロジを設定するには、一番下の各ブローカーに、メッセージを集中させている各ブローカーへのネットワークコネクターを含めます。

Broker3 のネットワークコネクター:

```
<networkConnectors>
    <networkConnector name="3_to_1" userName="myCommonUser" duplex="false"
        uri="static:(ssl://b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-
east-2.amazonaws.com:61617)"/>
    <networkConnector name="3_to_2" userName="myCommonUser" duplex="false"
        uri="static:(ssl://b-9876l5k4-32ji-109h-8gfe-7d65c4b132a1-2.mq.us-
east-2.amazonaws.com:61617)"/>
</networkConnectors>
```

Broker4 のネットワークコネクター:

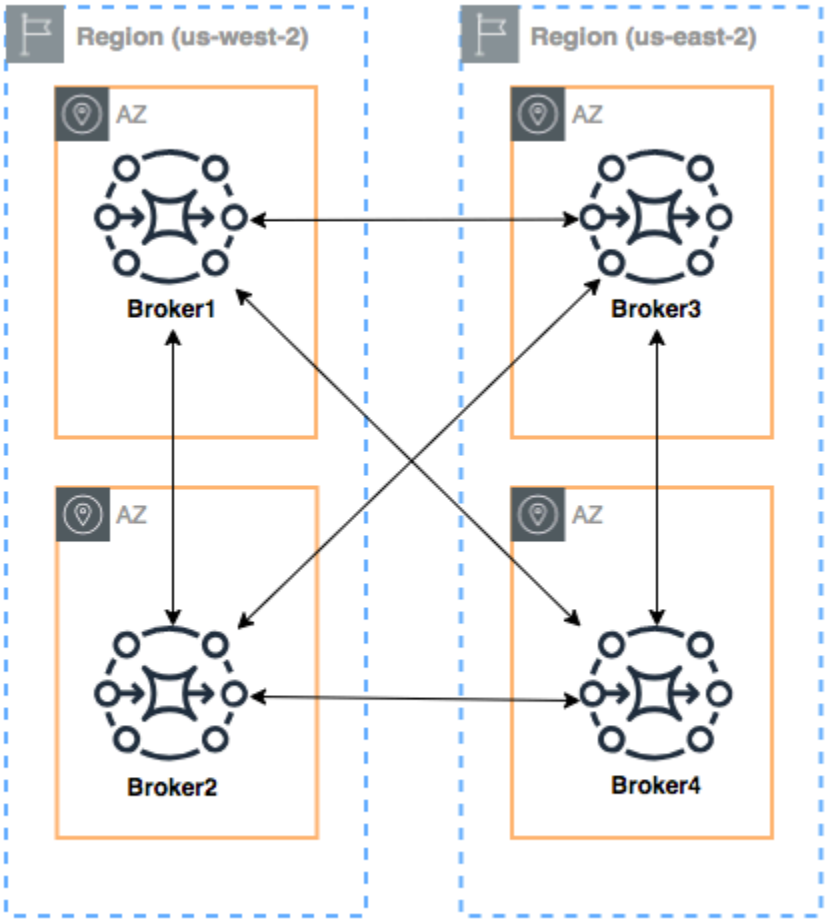
```
<networkConnectors>
    <networkConnector name="4_to_1" userName="myCommonUser" duplex="false"
        uri="static:(ssl://b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-
east-2.amazonaws.com:61617)"/>
    <networkConnector name="4_to_2" userName="myCommonUser" duplex="false"
        uri="static:(ssl://b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-
east-2.amazonaws.com:61617)"/>
</networkConnectors>
```

Broker5 のネットワークコネクター:

```
<networkConnectors>
    <networkConnector name="5_to_1" userName="myCommonUser" duplex="false"
        uri="static:(ssl://b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-
east-2.amazonaws.com:61617)"/>
    <networkConnector name="5_to_2" userName="myCommonUser" duplex="false"
        uri="static:(ssl://b-9876l5k4-32ji-109h-8gfe-7d65c4b132a1-2.mq.us-
east-2.amazonaws.com:61617)"/>
</networkConnectors>
```

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にネットワークコネクターを設定します。



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-
east-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-
east-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。

トピック

- Spring XML 設定ファイルの使用
- ActiveMQ ブローカー設定の作成、編集、適用
- Amazon MQ 設定で許可されている要素
- Amazon MQ 設定で許可されている要素とその属性
- 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

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-5.17.6.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。

ActiveMQ ブローカー設定の作成、編集、適用

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。

詳細については、次を参照してください。

- 構成
- Amazon MQ ブローカー設定のライフサイクル
- Amazon MQ Broker Configuration Parameters

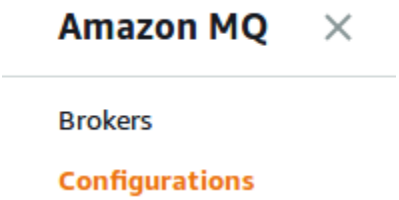
以下の例では、 AWS Management Consoleを使用して Amazon MQ ブローカーの設定を作成し、適用する方法を説明します。

トピック

- 新しい設定の作成
- 新しい設定リビジョンの作成
- 設定リビジョンをブローカーに適用する
- 設定のリビジョンの編集

新しい設定の作成

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. [Create configuration] (設定の作成) ページの [Details] (詳細) セクションで [Configuration name] (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

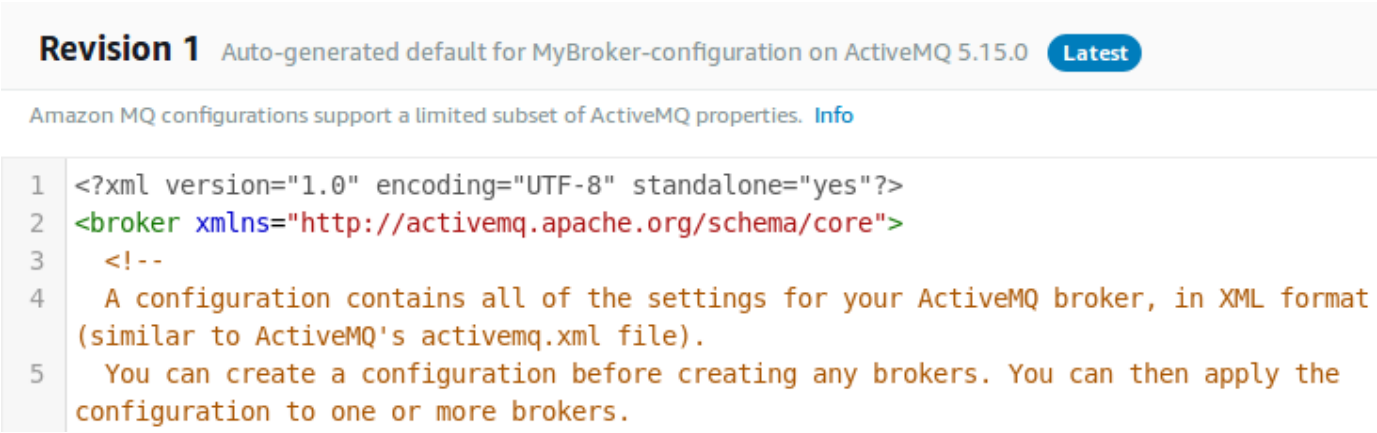
1. 設定リストから、 を選択します***MyConfiguration***。

> **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 ページには、新しい設定リビジョンが使用するブローカーエンジンの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。

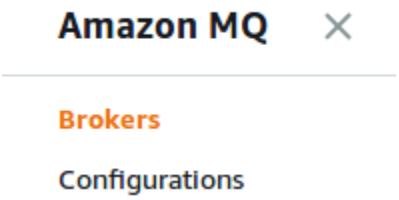
   設定の新しいリビジョンが保存されます。

> **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。
> 設定を変更しても、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。詳細については、「Amazon MQ ブローカー設定のライフサイクル」を参照してください。

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

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 (などMyBroker) を選択し、編集 を選択します。
3. 「編集***MyBroker***」ページの「設定」セクションで、「設定」と「リビジョン」を選択し、「変更のスケジュール」を選択します。
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、ブローカーはオフラインになります。

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。

設定のリビジョンの編集

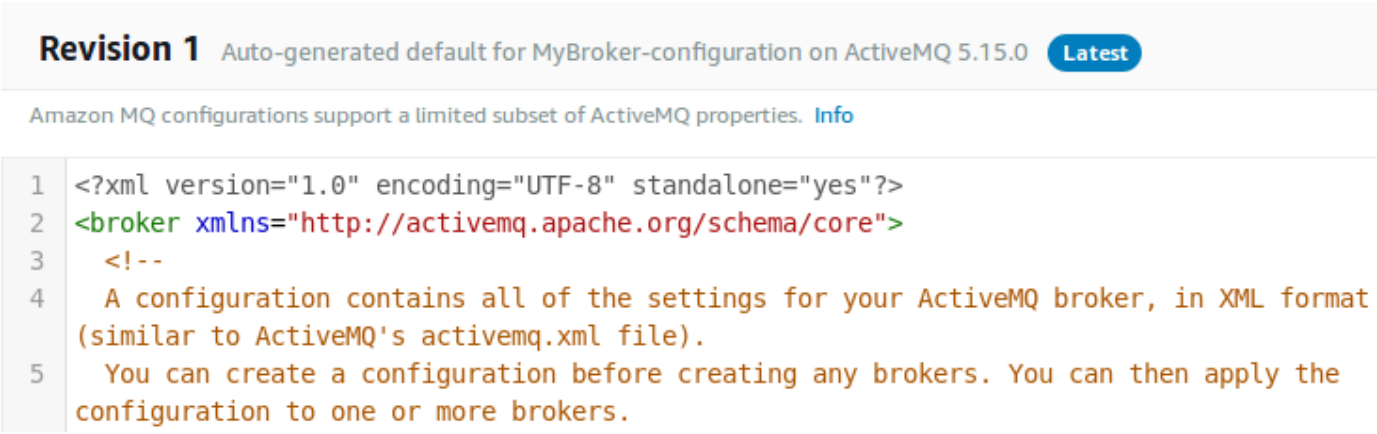
1. Amazon MQ コンソールにサインインします。
2. ブローカーリストからブローカー (などMyBroker) を選択し、編集 を選択します。
3. ***MyBroker*** ページで、編集 を選択します。
4. 「編集***MyBroker***」ページの「設定」セクションで、「設定」と「リビジョン」を選択し、「編集」を選択します。

> **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 ページには、設定が使用するブローカーエンジンの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。
> 設定を変更しても、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。詳細については、「Amazon MQ ブローカー設定のライフサイクル」を参照してください。
> 現在、設定を削除することはできません。

Amazon MQ 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。

要素
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)

要素
discarding (属性)
discardingDLQBrokerPlugin (属性)
fileCursor
fileDurableSubscriberCursor
fileQueueCursor
filteredDestination (属性)
fixedCountSubscriptionRecoveryPolicy (属性)
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

要素
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)
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)

要素
topic (属性)
transportConnector (属性)
uniquePropertyMessageEvictionStrategy (属性)
virtualDestinationInterceptor (子コレクション要素)
virtualTopic (属性)
vmCursor
vmDurableCursor
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。

要素	属性
abortSlowAckConsumerStrategy	abortConnection
	checkPeriod
	ignoreIdleConsumers
	ignoreNetworkConsumers
	maxSlowCount
	maxSlowDuration
	maxTimeSinceLastAck
	name

要素	属性
abortSlowConsumerStrategy	abortConnection
	checkPeriod
	ignoreNetworkConsumers
	maxSlowCount
	maxSlowDuration
	name
authorizationEntry	admin
	queue
	read
	tempQueue
	tempTopic
	topic
	write
broker	advisorySupport
	allowTempAutoCreationOnSend
	cacheTempDestinations
	consumerSystemUsagePortion
	dedicatedTaskRunner
	deleteAllMessagesOnStartup
	keepDurableSubsActive

要素	属性
	enableMessageExpirationOnActiveDurableSubs
	maxPurgedDestinationsPerSweep
	maxSchedulerRepeatAllowed
	monitorConnectionSplits
	networkConnectorStartAsync
	offlineDurableSubscriberTaskSchedule
	offlineDurableSubscriberTimeout
	persistenceThreadPriority
	persistent
	populateJMSXUserID
	producerSystemUsagePortion
	rejectDurableConsumers
	rollbackOnlyOnAsyncException
	schedulePeriodForDestinationPurge
	schedulerSupport
	splitSystemUsageForProducersConsumers
	taskRunnerPriority
	timeBeforePurgeTempDestinations

要素	属性
	useAuthenticatedPrincipalFo rJMSXUserID
	useMirroredQueues
	useTempMirroredQueues
	useVirtualDestSubs
	useVirtualDestSubsOnCreation
	useVirtualTopics
cachedMessageGroupMapFactory	cacheSize
compositeQueue	concurrentSend
	copyMessage
	forwardOnly
	name
	sendWhenNotMatched
compositeTopic	concurrentSend
	copyMessage
	forwardOnly
	name
	sendWhenNotMatched
条件付きNetworkBridgeFilterFactory	rateDuration
	rateLimit
	replayDelay

要素	属性
	replayWhenNoConsumers
	selectorAware 以下でサポート Apache ActiveMQ 5.16.x
constantPendingMessageLimitStrategy	limit
discarding	deadLetterQueue
	enableAudit
	expiration
	maxAuditDepth
	maxProducersToAudit
	processExpired
	processNonPersistent
discardingDLQBrokerPlugin	dropAll
	dropOnly
	dropTemporaryQueues
	dropTemporaryTopics
	reportInterval
filteredDestination	queue
	selector

要素	属性
	topic
fixedCountSubscriptionRecoveryPolicy	maximumSize
fixedSizedSubscriptionRecoveryPolicy	maximumSize
	useSharedBuffer
forcePersistencyModeBrokerPlugin	persistenceFlag
individualDeadLetterStrategy	destinationPerDurableSubscriber
	enableAudit
	expiration
	maxAuditDepth
	maxProducersToAudit
	processExpired
	processNonPersistent
	queuePrefix
	queueSuffix
	topicPrefix
	topicSuffix
	useQueueForQueueMessages
	useQueueForTopicMessages
messageGroupHashBucketFactory	bucketCount
	cacheSize

要素	属性
mirroredQueue	copyMessage
	postfix
	prefix
oldestMessageEvictionStrategy	evictExpiredMessagesHighWat ermark
oldestMessageWithLowestPrio rityEvictionStrategy	evictExpiredMessagesHighWat ermark
policyEntry	advisoryForConsumed
	advisoryForDelivery
	advisoryForDiscardingMessages
	advisoryForFastProducers
	advisoryForSlowConsumers
	advisoryWhenFull
	allConsumersExclusiveByDefault
	alwaysRetroactive
	blockedProducerWarningInterval
	consumersBeforeDispatchStarts
	cursorMemoryHighWaterMark
	doOptimzeMessageStorage
	durableTopicPrefetch
	enableAudit

要素	属性
	expireMessagesPeriod
	gcInactiveDestinations
	gcWithNetworkConsumers
	inactiveTimeoutBeforeGC
	inactiveTimoutBeforeGC
	includeBodyForAdvisory
	lazyDispatch
	maxAuditDepth
	maxBrowsePageSize
	maxDestinations
	maxExpirePageSize
	maxPageSize
	maxProducersToAudit
	maxQueueAuditDepth
	memoryLimit
	messageGroupMapFactoryType
	minimumMessageSize
	optimizedDispatch
	optimizeMessageStoreInFligh tLimit
	persistJMSRedelivered

要素	属性
	prioritizedMessages
	producerFlowControl
	queue
	queueBrowserPrefetch
	queuePrefetch
	reduceMemoryFootprint
	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

要素	属性
	useConsumerPriority
usePrefetchExtension	
prefetchRatePendingMessageLimitStrategy	multiplier
queryBasedSubscriptionRecoveryPolicy	query
queue	DLQ
	physicalName
redeliveryPlugin	fallbackToDeadLetter
	sendToDlqIfMaxRetriesExceeded
redeliveryPolicy	backOffMultiplier
	collisionAvoidancePercent
	initialRedeliveryDelay
	maximumRedeliveries
	maximumRedeliveryDelay
	preDispatchCheck
	queue
	redeliveryDelay
	tempQueue
	tempTopic
	topic

要素	属性
	useCollisionAvoidance
	useExponentialBackOff
sharedDeadLetterStrategy	enableAudit
	expiration
	maxAuditDepth
	maxProducersToAudit
	processExpired
	processNonPersistent
storeDurableSubscriberCursor	immediatePriorityDispatch
	useCache
tempDestinationAuthorizationEntry	admin
	queue
	read
	tempQueue
	tempTopic
	topic
	write
tempQueue	DLQ
	physicalName
tempTopic	DLQ

要素	属性
	physicalName
timedSubscriptionRecoveryPolicy	zeroExpirationOverride
timeStampingBrokerPlugin	recoverDuration
	futureOnly
	processNetworkMessages
	ttlCeiling
topic	DLQ
	physicalName
transportConnector	•
	name
	updateClusterClients
	rebalanceClusterClients
	updateClusterClientsOnRemove
uniquePropertyMessageEvictionStrategy	evictExpiredMessagesHighWatermark
	propertyName
virtualTopic	concurrentSend
	local
	dropOnResourceLimit
	name

要素	属性
	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

ネットワークConnectionStart非同期

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。

要素	子コレクション要素	子要素
authorizationMap	authorizationEntries	authorizationEntry
		tempDestinationAuthorizationEntry
	defaultEntry	authorizationEntry
		tempDestinationAuthorizationEntry
	tempDestinationAuthorizationEntry	tempDestinationAuthorizationEntry
authorizationPlugin	map	authorizationMap
broker	destinationInterceptors	mirroredQueue
		virtualDestinationInterceptor
	destinationPolicy	policyMap
	destinations	queue
		tempQueue
		tempTopic
		topic
	networkConnectors	networkConnector

要素	子コレクション要素	子要素
	persistenceAdapter	kahaDB
	plugins	authorizationPlugin
		discardingDLQBroke rPlugin
		forcePersistencyMo deBrokerPlugin
		redeliveryPlugin
		statisticsBrokerPl ugin
		timeStampingBroker Plugin
	systemUsage	systemUsage
	transportConnector	name
		updateClusterClients
		rebalanceClusterCl ients
		updateClusterClien tsOnRemove
compositeQueue	forwardTo	queue
		tempQueue
		tempTopic
		topic
		filteredDestination

要素	子コレクション要素	子要素
compositeTopic	forwardTo	queue
		tempQueue
		tempTopic
		topic
		filteredDestination
policyEntry	deadLetterStrategy	discarding
		individualDeadLetterStrategy
		sharedDeadLetterStrategy
	destination	queue
		tempQueue
		tempTopic
		topic
	dispatchPolicy	priorityDispatchPolicy
		priorityNetworkDispatchPolicy
		roundRobinDispatchPolicy
		simpleDispatchPolicy
		strictOrderDispatchPolicy

要素	子コレクション要素	子要素
		clientIdFilterDispatchPolicy
	messageEvictionStrategy	oldestMessageEvictionStrategy
		oldestMessageWithLowestPriorityEvictionStrategy
		uniquePropertyMessageEvictionStrategy
	messageGroupMapFactory	cachedMessageGroupMapFactory
		messageGroupHashBucketFactory
		simpleMessageGroupMapFactory
	pendingDurableSubscriberPolicy	fileDurableSubscriberCursor
		storeDurableSubscriberCursor
		vmDurableCursor
	pendingMessageLimitStrategy	constantPendingMessageLimitStrategy
		prefetchRatePendingMessageLimitStrategy
	pendingQueuePolicy	fileQueueCursor

要素	子コレクション要素	子要素
		storeCursor
		vmQueueCursor
	pendingSubscriberPolicy	fileCursor
		vmCursor
	slowConsumerStrategy	abortSlowAckConsumerStrategy
		abortSlowConsumerStrategy
	subscriptionRecoveryPolicy	fixedCountSubscriptionRecoveryPolicy
		fixedSizedSubscriptionRecoveryPolicy
		lastImageSubscriptionRecoveryPolicy
		noSubscriptionRecoveryPolicy
		queryBasedSubscriptionRecoveryPolicy
		retainedMessageSubscriptionRecoveryPolicy
timedSubscriptionRecoveryPolicy		
policyMap	defaultEntry	policyEntry

要素	子コレクション要素	子要素
	policyEntries	policyEntry
redeliveryPlugin	redeliveryPolicyMap	redeliveryPolicyMap
redeliveryPolicyMap	defaultEntry	redeliveryPolicy
	redeliveryPolicyEntries	redeliveryPolicy
retainedMessageSubscriptionRecoveryPolicy	wrapped	fixedCountSubscriptionRecoveryPolicy
		fixedSizedSubscriptionRecoveryPolicy
		lastImageSubscriptionRecoveryPolicy
		noSubscriptionRecoveryPolicy
		queryBasedSubscriptionRecoveryPolicy
		retainedMessageSubscriptionRecoveryPolicy
		timedSubscriptionRecoveryPolicy
sharedDeadLetterStrategy	deadLetterQueue	queue
		tempQueue
		tempTopic
		topic

要素	子コレクション要素	子要素
virtualDestination Interceptor	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
- networkConnector
- kahaDB
- systemUsage

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。

[Default] (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>
```

networkConnector

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

トピック

- 属性
- 設定例

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。

[Default] (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。

[Default] (デフォルト): `false`

name

ブローカーのネットワークのブリッジの名前。

[Default] (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。

[Default] (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。

[Default] (デフォルト): `null`

設定例

> **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-east-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。

[Default] (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。

[Default] (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。

[Default] (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。

[Default] (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。

[Default] (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。

[Default] (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。

[Default] (デフォルト): false

sendFailIfNoSpaceAfterTimeout

[Default] (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 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示し、 はパッチバージョン番号Z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更された場合、バージョン変更はマイナーと見なされます。例えば、バージョン 5.17 から 5.18 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 ActiveMQ では、すべてのブローカーが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の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の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 でサポートされてい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には、ブローカーエンジンバージョンがサポート終了になる時期が表示されます。バージョンがサポート終了になると、Amazon MQ は、このバージョンのすべてのブローカーを次にサポートされているバージョンに自動的にアップグレードします。Amazon MQ は、バージョンがサポートを終了する少なくとも 90 日前に通知します。

Apache ActiveMQ バージョン	Amazon MQ のサポート終了
ActiveMQ 5.17 (推奨)	
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。を使用してブローカー AWS Management Console を作成する場合、Amazon MQ は自動的に最新のエンジンバージョン番号にデフォルト設定されます。 AWS CLI または Amazon MQ API を使用してブローカーを作成する場合は、エンジンのバージョン番号が必要です。バージョン番号を指定しない場合は、操作で例外が発生します。詳細については、AWS CLI コマンドリファレンスの「`create-broker`」、および Amazon MQ REST API リファレンスの「`CreateBroker`」を参照してください。

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、次にサポートされているメジャー、マイナー、またはパッチ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 中にブローカーをサポートされている最新のパッチバージョンにアップグレードします。 Amazon MQ

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

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。

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。

- この Java コード OpenWire の例では、ブローカーに接続し、キューを作成し、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

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、ブローカーの名前を選択します (例: MyBroker)。
3. ***MyBroker*** ページの Connections セクションで、ブローカーのウェブコンソール URL とワイヤレベルのプロトコルのアドレスとポートを書き留めます。
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);
        }
    }
    }
```

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

- ActiveMQ ブローカーの作成と設定
- 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

- ActiveMQ ブローカーユーザーの作成と管理

ActiveMQ ブローカーの作成と設定

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`、`t3`) およびサイズ (`large`、`micro`) を組み合わせた説明がブローカーインスタンスタイプ (`mq.m5.large` など) になります。詳細については、「ブローカー」を参照してください。

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 Management Consoleを使用したブローカーを作成および設定する方法を説明します。

トピック

- ステップ 1: ブローカーの基本設定を定義する
- ステップ 2: (オプション) ブローカーの追加設定を定義する
- ステップ 3: ブローカー作成の完了
- ブローカーエンジンのバージョン、インスタンスタイプ、 CloudWatch ログ、メンテナンス設定の編集

ステップ 1: ブローカーの基本設定を定義する

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、「Broker Architecture」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、「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、「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。
- ブローカーのネットワークのサンプル設計図の詳細については、「サンプル設計図」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。ブローカー名には、 CloudWatch ログを含む他の 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。

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。ブローカーユーザー名は、 CloudWatch ログを含む他の 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。

ステップ 2: (オプション) ブローカーの追加設定を定義する

> **Important**
>
> - サブネット – 単一インスタンスブローカーには、1 つのサブネットが必要です (デフォルトのサブネットなど)。アクティブ/スタンバイブローカーには、2 つのサブネットが必要です。
> - セキュリティグループ – 単一インスタンスブローカーとアクティブ/スタンバイブローカーのどちらにも、少なくとも 1 つのセキュリティグループが必要です (デフォルトのセキュリティグループなど)。
> - 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。EC2-Classic リソースはサポートされていません。Amazon MQ はデフォルトの VPC テナンシーのみをサポートしており、専用の VPC テナンシーはサポートしていません。
> - 暗号化 – データを暗号化するカスタマーマスターキーを選択します。「保管中の暗号化」を参照してください。
> - パブリックアクセシビリティ – パブリックアクセシビリティを無効にすると、ブローカーにアクセスできるのは VPC 内のみになります。詳細については、「パブリックアクセシビリティのないブローカーを優先する」および「パブリックアクセシビリティが無効化されたブローカーウェブコンソールへのアクセス」を参照してください。

1. [詳細設定] セクションを展開します。
2. [設定] セクションで、[Create a new configuration with default values (デフォルト値を使用して新しい設定を作成する)] または [Select an existing configuration (既存の設定を選択する)] を選択

します。詳細については、「構成」および「Amazon MQ Broker Configuration Parameters」を参照してください。

3. ログセクションで、一般ログと監査ログを Amazon CloudWatch Logs に発行するかどうかを選択します。詳細については、「Configuring Amazon MQ to publish logs to Amazon CloudWatch Logs」を参照してください。

> **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。Amazon MQ のリソースベースのポリシーを設定しない場合、ブローカーはログを CloudWatch Logs に発行できません。

4. [Network and security section] (ネットワークおよびセキュリティセクション) で、ブローカーの接続を設定します。

   a. 次のいずれかを行います。

      - [Use the default VPC and subnet(s)]、[Use the default security group(s)] (デフォルトの VPC とサブネットを使用、デフォルトのセキュリティグループを使用) を選択します。
      - [Select existing VPC and subnet(s)]、[Select existing security group(s)] (既存の VPC とサブネットを選択する、既存のセキュリティグループの選択) を選択します。
        1. このオプションを選択すると、Amazon VPC コンソールで新しい Virtual Private Cloud (VPC) を作成、既存の VPC を選択、またはデフォルトの VPC を選択することができます。詳細については、Amazon VPC ユーザーガイドの「Amazon VPC とは」を参照してください。
        2. VPC を作成または選択したら、Amazon VPC コンソールで新しい [Subnet(s)] (サブネット) を作成するか、既存のサブネットを選択できます。詳細については、Amazon VPC ユーザーガイドの「VPC とサブネット」を参照してください。
        3. サブネットを作成または選択すると、[セキュリティグループ] を選択できます。

   b. データの暗号化に使用されるカスタマーマスターキー (CMK) を選択します。保管中の暗号化 を参照してください。

   c. ブローカーの [パブリックアクセシビリティ] を選択します。

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。

a. Apache からの新しいバージョンのリリースに伴ってブローカーをアップグレードするには、[Enable automatic minor version upgrades] (自動マイナーバージョンアップグレードの有効化) 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。

> **Note**
>
> アクティブ/スタンバイブローカーについては、ブローカーインスタンスのいずれかで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。

b. 以下のいずれかを実行します。

- Amazon MQ がメンテナンスウィンドウを自動的に選択できるようにするには、[No preference] (指定なし) を選択します。
- カスタムのメンテナンスウィンドウを設定するには、[Select maintenance window (メンテナンスウィンドウの選択)] を選択し、アップグレードの [Start day (開始日)] と [Start time (開始時刻)] を指定します。

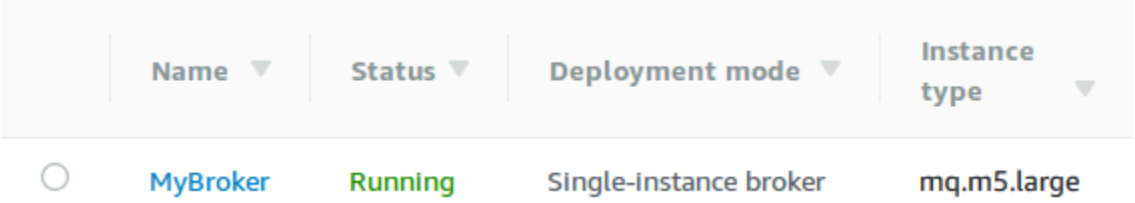
ステップ 3: ブローカー作成の完了

1. [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。



2. を選択します***MyBroker***。

MyBroker ページの Connect セクションで、ブローカーの ActiveMQ ウェブコンソール URL を書き留めます。次に例を示します。

```
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。 OpenWire エンドポイントの例を次に示します。

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

> **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、「Broker Architecture」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できます。

ブローカーエンジンのバージョン、インスタンスタイプ、 CloudWatch ログ、メンテナンス設定の編集

ブローカー設定の編集と設定リビジョンの管理に加えて、ブローカーに固有の設定を行うことができます。

> **Note**
>
> 自動マイナーバージョンのアップグレード以外のすべての設定では、変更をスケジュールする必要があります。詳細については、「Amazon MQ ブローカー設定のライフサイクル」を参照してください。

以下の例では、 AWS Management Consoleを使用して Amazon MQ ActiveMQ ブローカーの設定を編集する方法を説明します。

ActiveMQ 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 (などMyBroker) を選択し、編集 を選択します。
3. 編集***MyBroker***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. [設定] セクションでブローカーの設定とリビジョンを選択します。詳細については、「Creating and applying broker configurations」を参照してください。
5. [Security and network] (セキュリティとネットワーク) セクションで、[Security group(s)] (セキュリティグループ) ドロップダウンからグループを選択するか、[Create a new security group] (新しいセキュリティグループの作成) を選択して Amazon VPC コンソールを開きます。
6. CloudWatch ログセクションで、一般ログと監査ログを Amazon CloudWatch Logs に発行するかどうかを選択します。

   ActiveMQ ブローカーの CloudWatch ログ設定の詳細については、「」を参照してくださいConfiguring Amazon MQ to publish logs to Amazon CloudWatch Logs。

   > **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。
   > Amazon MQ のリソースベースのポリシーを設定しない場合、ブローカーはログを CloudWatch Logs に発行できません。

7. [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。

> **Note**
>
> アクティブ/スタンバイブローカーについては、ブローカーインスタンスのいずれかで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。

8. [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。

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。ブローカーのネットワークは、高可用性やスケーラビリティなどのアプリケーションのニーズに応じて、さまざまなトポロジ (コンセントレータ、ハブアンドスポーク、ツリー、またはメッシュなど) で設定できます。例えば、ブローカーのハブアンドスポークネットワークは耐障害性を高めることができ、1 つのブローカーが到達不能な場合にはメッセージを保存します。コンセントレータトポロジを使用するブローカーのネットワークは、多数の着信メッセージの負荷をより良く処理するために、着信メッセージを受け入れる多数のブローカーからメッセージを収集し、それらをより中核的なブローカーに集中させます。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。

- ブローカーの Amazon MQ 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`
- `######ConnectionStart###`

- 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
- ステップ 1: ブローカー間のトラフィックを許可する
- 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
- 次のステップ

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、「Creating and configuring a broker」 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と管理」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 ページの「詳細」セクションの「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 を選択します。

   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 には、 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. [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。

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. 「編集 MyBroker1」ページの「設定」セクションで、「 を表示」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する場合は、 uri のstatic:プレフィックスと OpenWire エンドポイントを使用しますMyBroker2。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する場合は、次のクエリパラメータ を指定して、両方のブローカーuriのstatic +failoverトランスポートと OpenWireエンドポイントを使用します?randomize=false&maxReconnectAttempts=0。例:

```
<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-
east-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. 「編集 MyBroker1」ページの「設定」セクションで、「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. [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 ウェブコンソールの場合) およびポート 61617 (エンドポイントの場合 OpenWire) MyBroker1で、ブローカーのローカルマシンからのインバウンド接続を有効にしていることを確認してください。
> 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。

   [Network Bridges] (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 [Remote Broker] (リモートブローカー) と [Remote Address] (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。
   例:

```
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 の OpenWire エンドポイントに接続MyBroker2し、キュー からのメッセージの使用を開始しますMyQueue。

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。例:

```
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 の OpenWire エンドポイントに接続MyBroker1し、キュー への永続メッセージの生成を開始しますMyQueue。

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

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、ブローカーの名前を選択します (例: MyBroker)。
3. ***MyBroker*** ページの Connections セクションで、ブローカーのウェブコンソール URL とワイヤレベルのプロトコルのアドレスとポートを書き留めます。
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

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、「Broker Architecture」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できます。

```
// 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

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**
>
> RabbitMQ ブローカーでは LDAP 統合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。

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 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 authentication and authorization] (LDAP 認証と認可) を選択します。

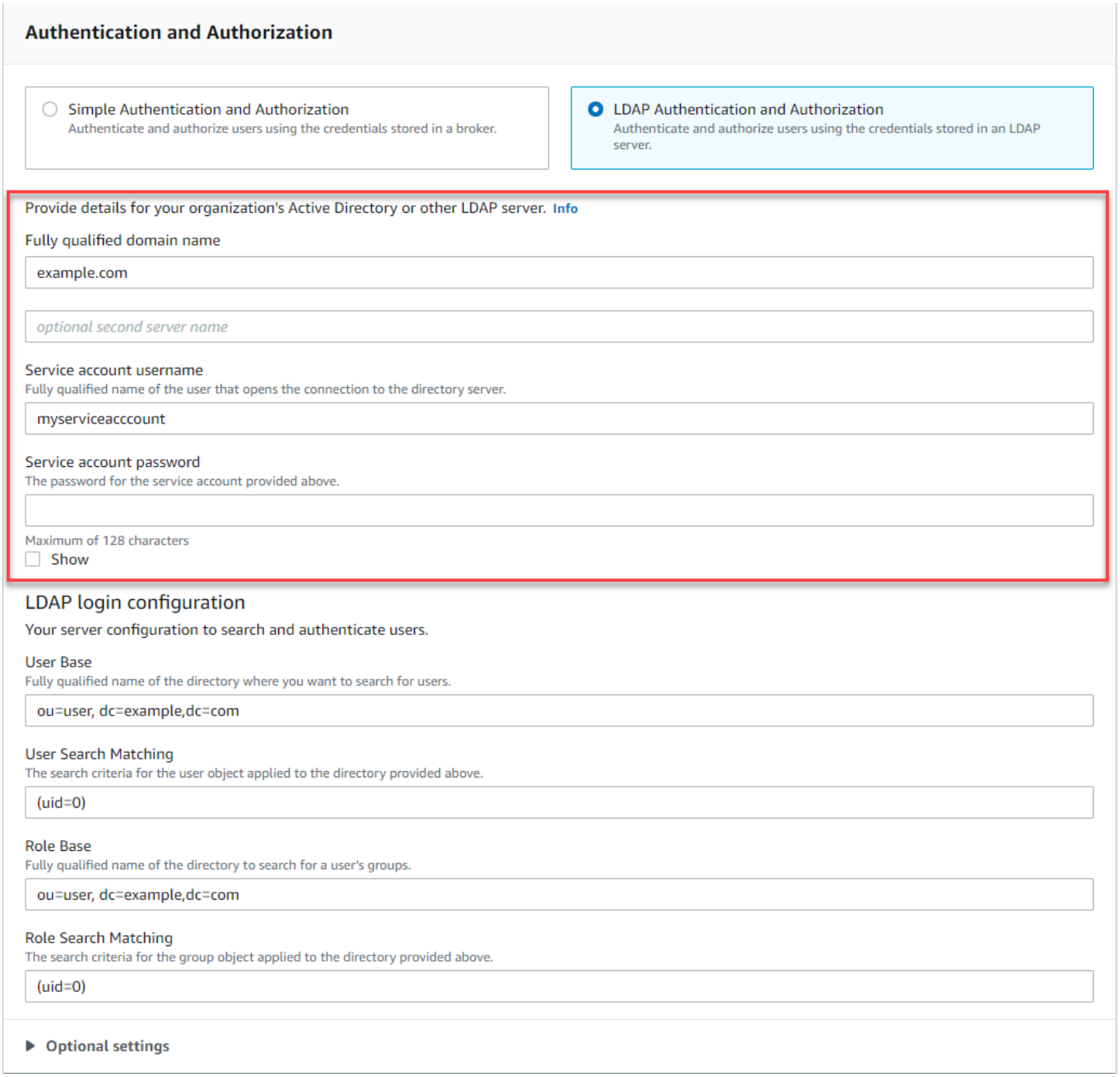
サービスアカウントに関する以下の情報を入力します。

- Fully qualified domain name (完全修飾ドメイン名) 認証リクエストと認可リクエストが発行される 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。

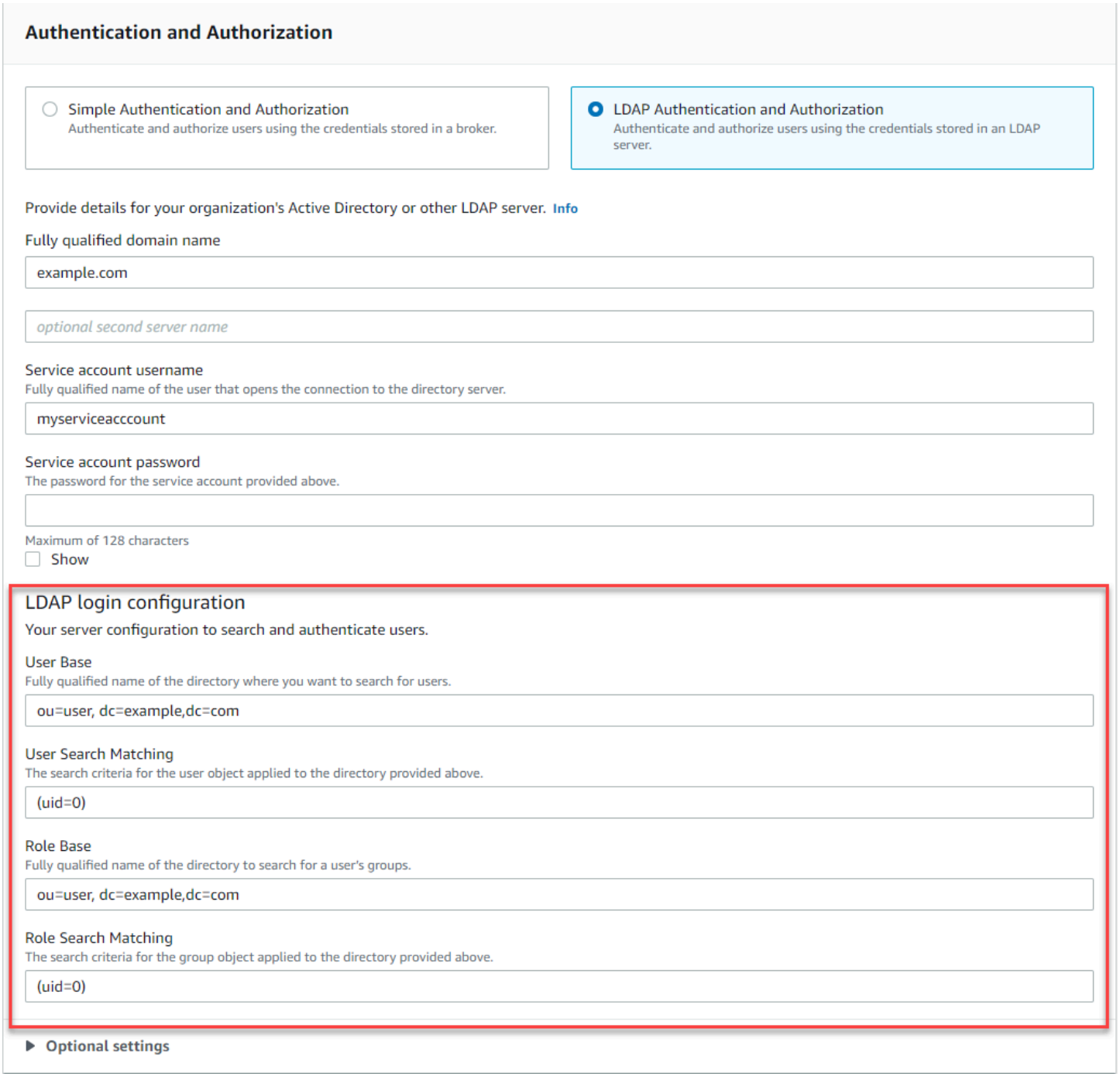


[LDAP login configuration] (LDAP ログイン設定) セクションで、以下の必須情報を入力します。

- 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 認可 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。

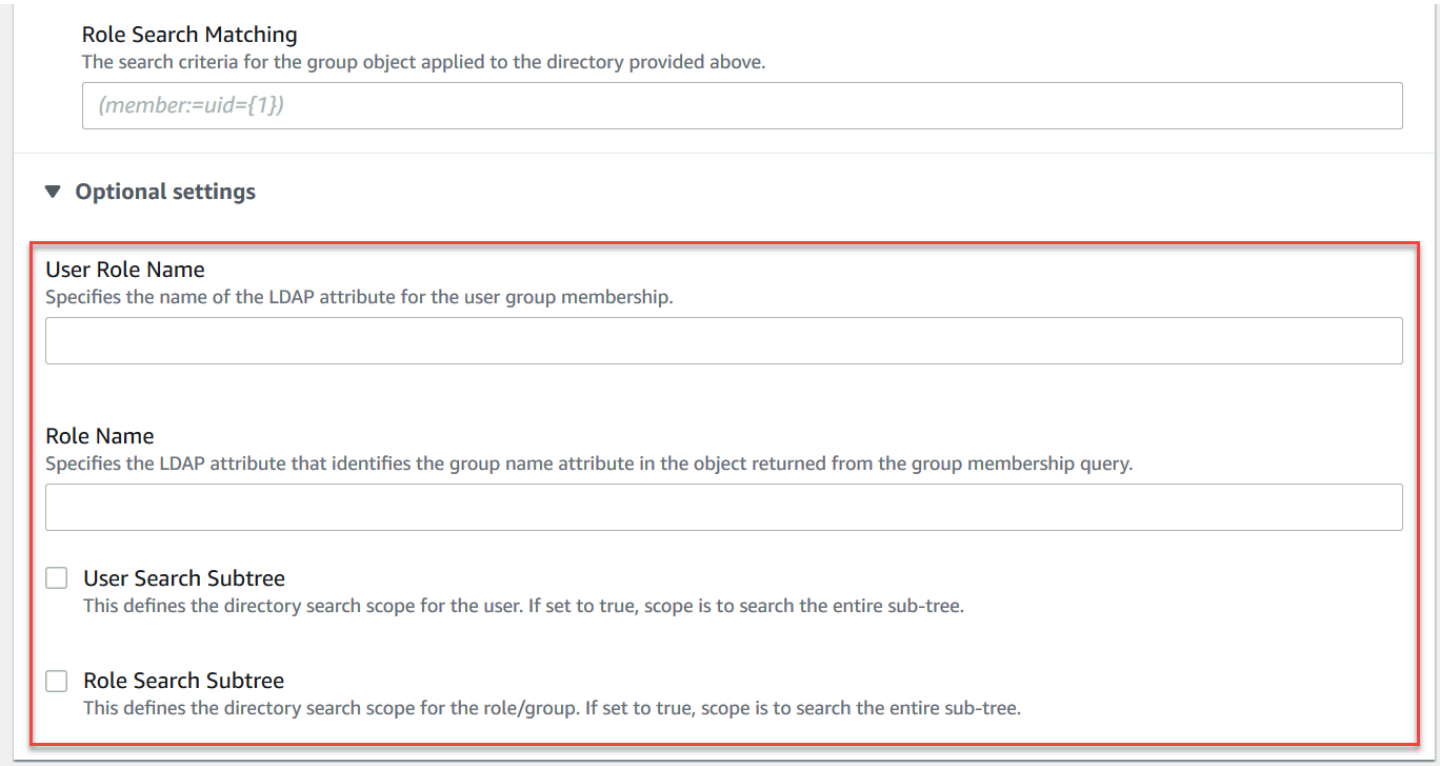
- User Role Name (ユーザーロール名) ユーザーのグループメンバーシップに関するユーザーのディレクトリエントリ内の LDAP 属性の名前です。場合によっては、ユーザーのディレクトリエントリ内の属性の値によって、ユーザーロールを識別できることもあります。`userRoleName` オプションは、この属性の名前を指定することを可能にします。例えば、以下のユーザーエントリについて考えてみましょう。

```
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。

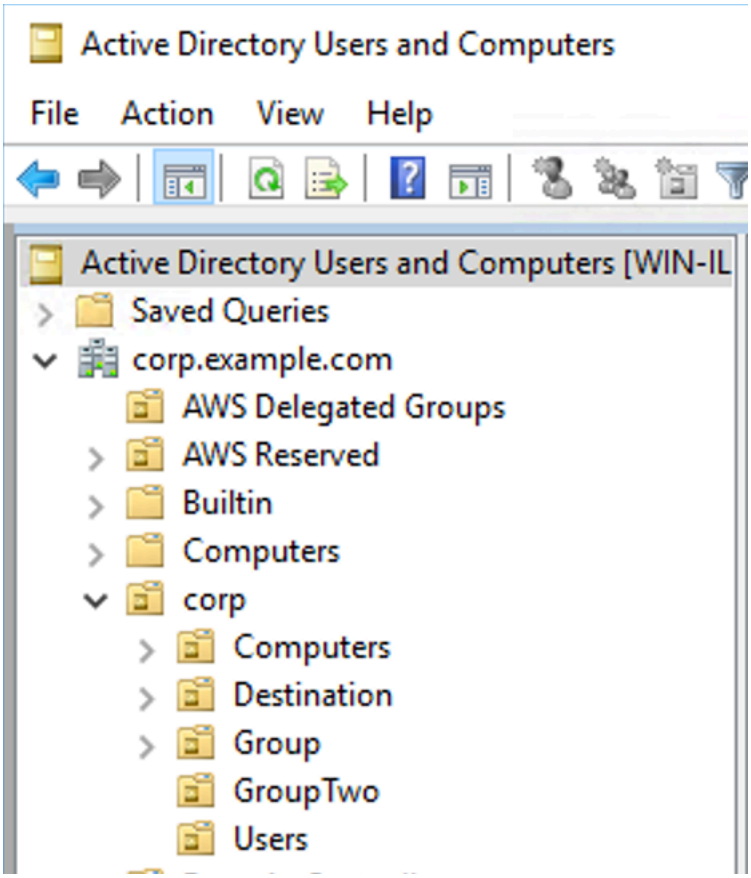
認証

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。例えば、AWS Managed Microsoft AD を使用していて、ドメインコンポーネント `corp`、`example`、および `com` があり、これらの中に組織単位 `corp` および `Users` がある場合は、以下をユーザーベースとして使用します。

```
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。

認可

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**
>
> ブローカーの `activemq.xml` 設定ファイルで `cachedLDAPAuthorizationMap` 要素を使用できるようにするには、AWS Management Console を使用して設定を作成するときに [LDAP Authentication and Authorization (LDAP 認証と認可)] オプションを選択するか、Amazon MQ API を使用して新しい設定を作成するときに `authenticationStrategy` プロパティを `LDAP` 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`
>
> ブローカーの作成時、Amazon MQ は上記の属性の代わりに、AWS Management Console経由で提供される値、または API リクエストの `ldapServerMetadata` プロパティに指定する値を使用します。

以下は、`cachedLdapAuthorizationMap` の実用例です。

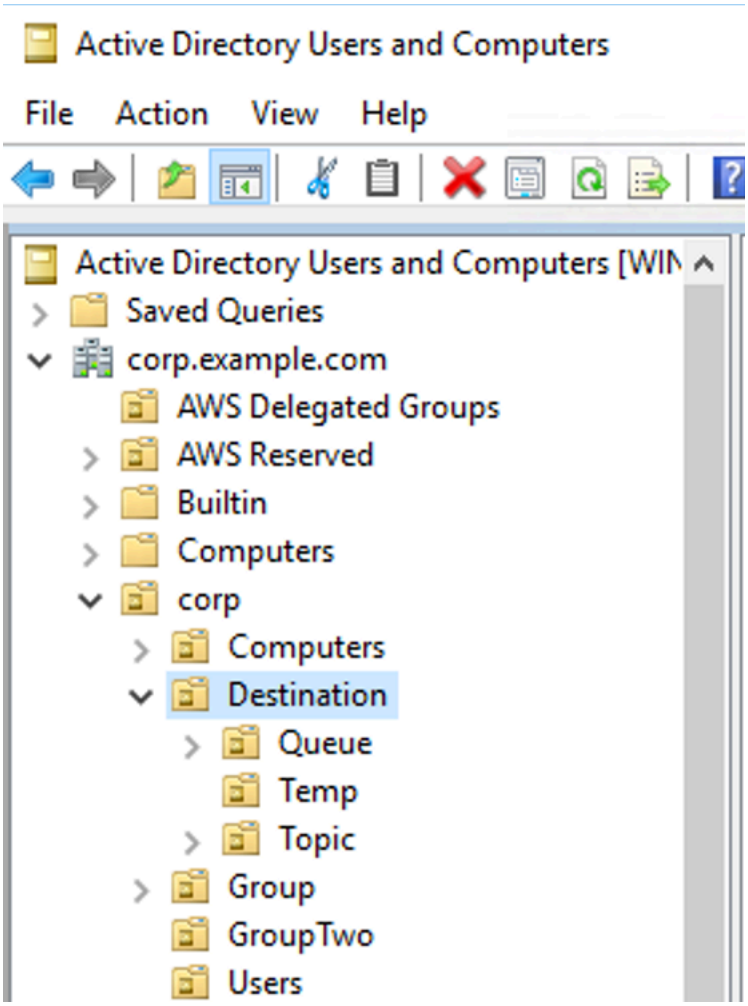
```
<authorizationPlugin>
    <map>
        <cachedLDAPAuthorizationMap
            queueSearchBase="ou=Queue,ou=Destination,ou=corp,dc=corp,dc=example,dc=com"
            topicSearchBase="ou=Topic,ou=Destination,ou=corp,dc=corp,dc=example,dc=com"
```

```
            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、同じ corp、example、および com ドメインコンポーネントを使用する、AWS Managed Microsoft AD での上記の例には、destination という名前の組織単位を指定して、すべての送信先タイプを含めます。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。

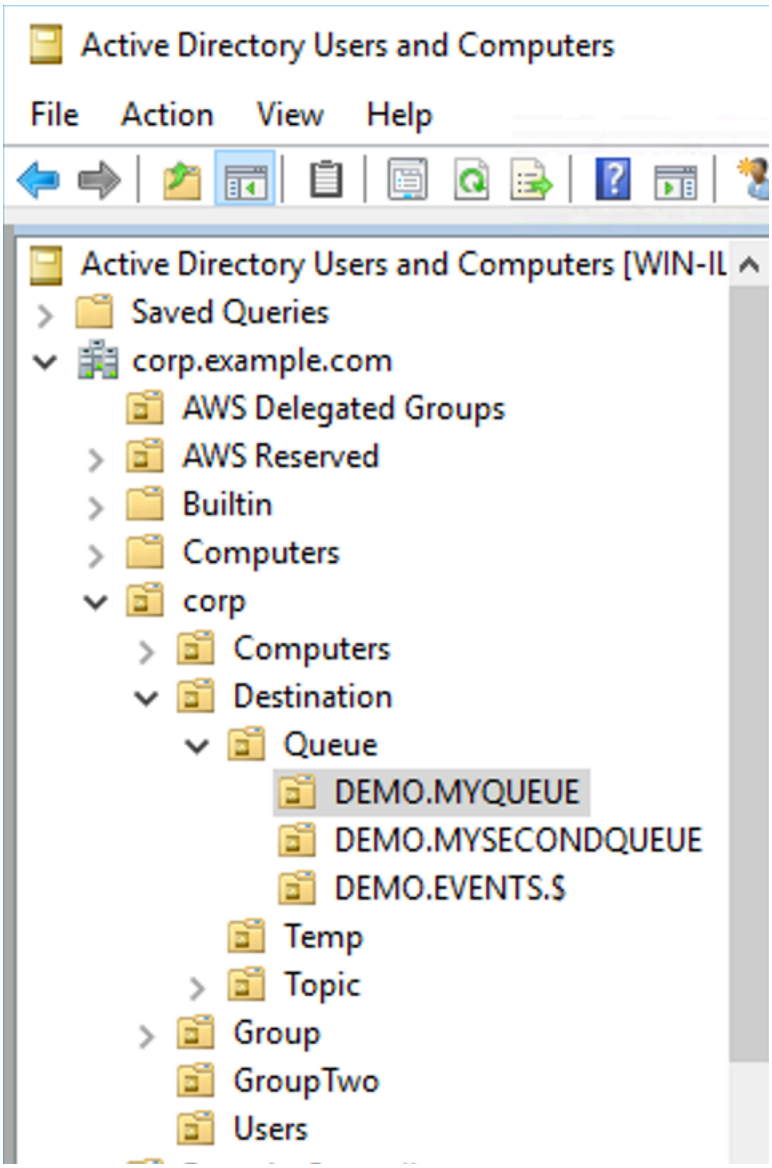
```
OU=DEMO.EVENTS.$,OU=Queue,OU=Destination,OU=corp,DC=corp,DC=example,DC=com
```

> **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。

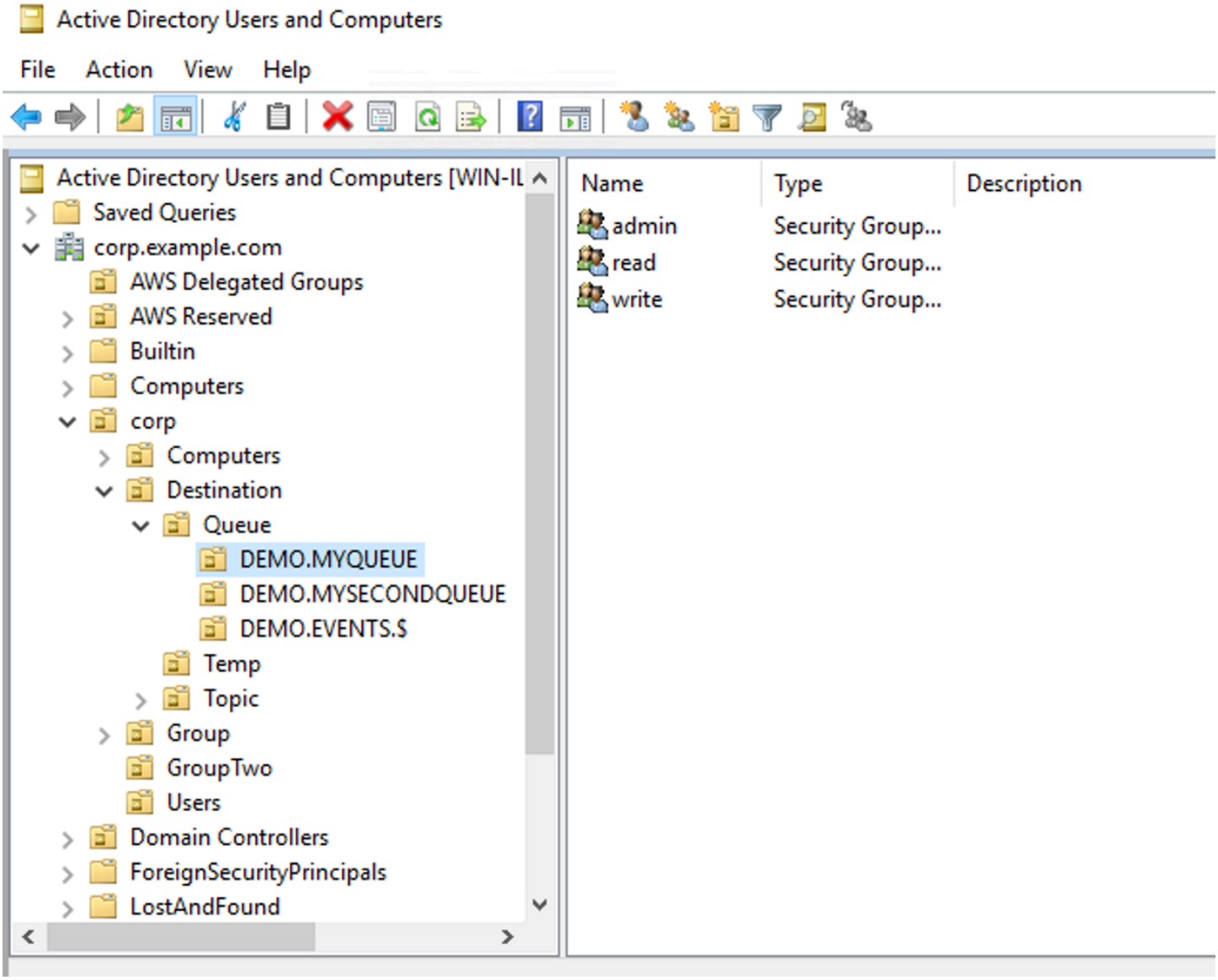
```
OU=DEMO.MYQUEUE,OU=Queue,OU=Destination,OU=corp,DC=corp,DC=example,DC=com
```



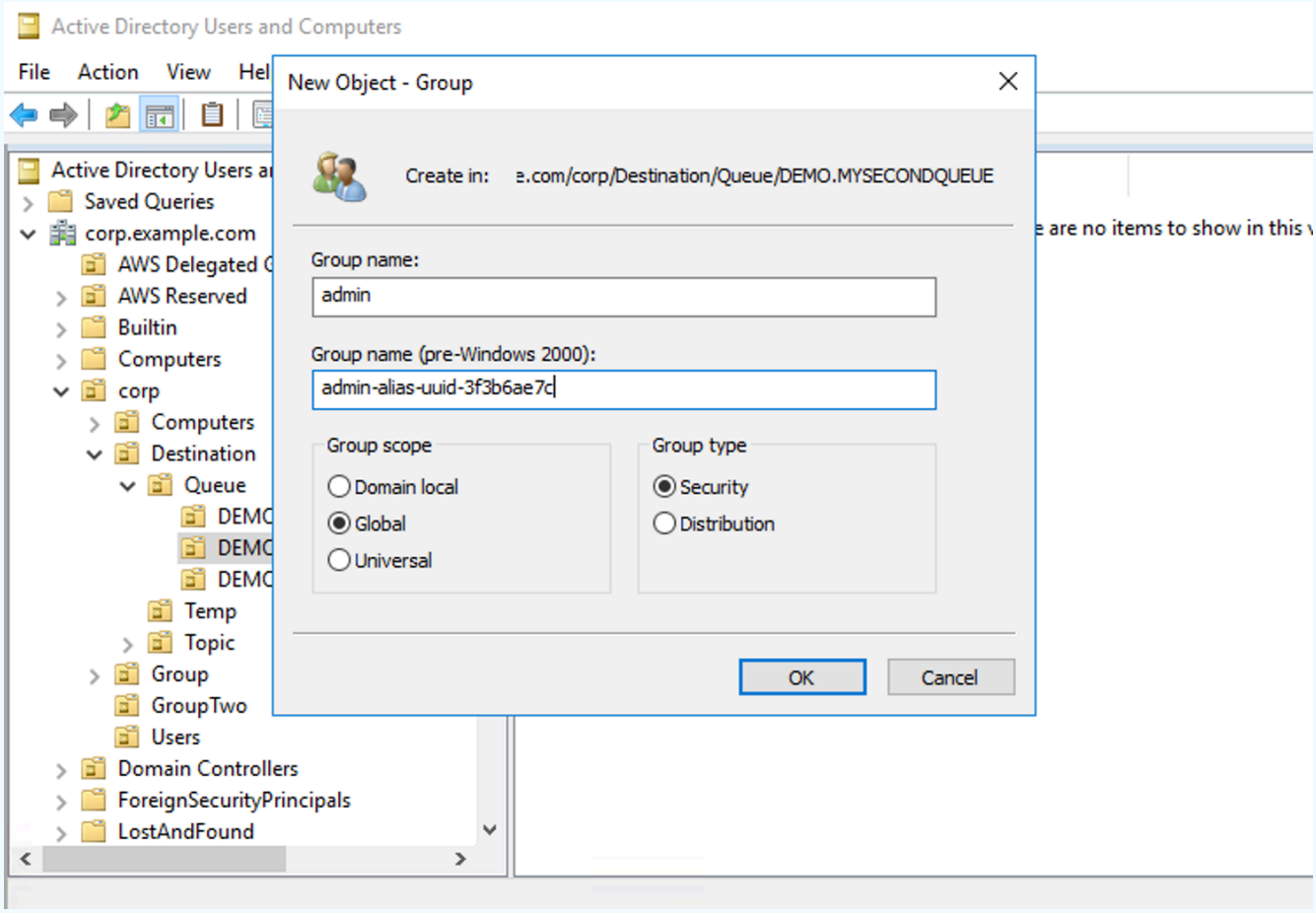
セキュリティグループ

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。

ActiveMQ ブローカーユーザーの作成と管理

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。

以下の例では、 AWS Management Console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。

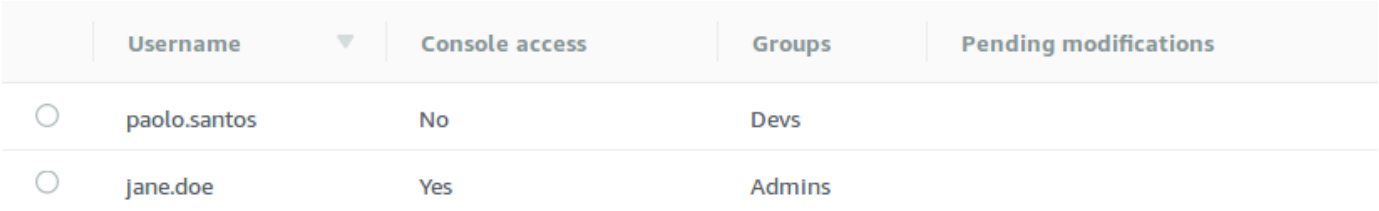
トピック

- 新しいユーザーを作成する
- 既存のユーザーを編集するには
- 既存のユーザーを削除するには

新しいユーザーを作成する

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の名前 (例: MyBroker) を選択し、詳細の表示 を選択します。

   MyBroker ページの「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一覧表示されます。



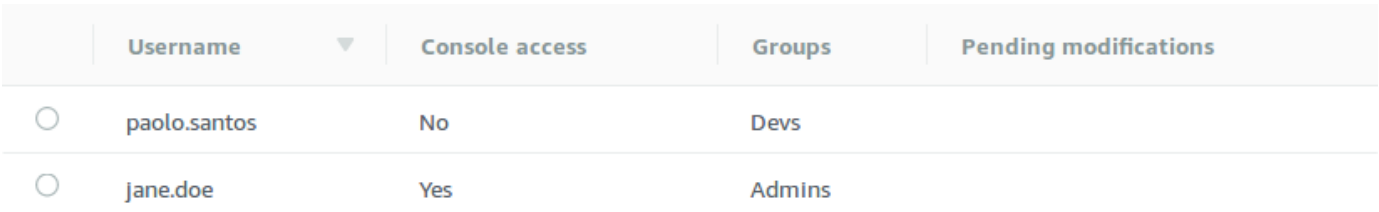
3. [ユーザーの作成] を選択します。
4. [ユーザーの作成] ダイアログボックスに、[ユーザー名] と [パスワード] を入力します。
5. (省略可能) ユーザーが属するグループの名前をコンマで区切って入力します (例: `Devs,Admins`)。
6. (省略可能) ユーザーが ActiveMQ ウェブコンソールにアクセスできるようにするには、[ActiveMQ ウェブコンソール] を選択します。
7. [Create user] (ユーザーの作成) をクリックします。

> **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。詳細については、「Amazon MQ ブローカー設定のライフサイクル」を参照してください。

既存のユーザーを編集するには

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 (例: MyBroker) を選択し、詳細の表示 を選択します。

   MyBroker ページの「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一覧表示されます。



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。

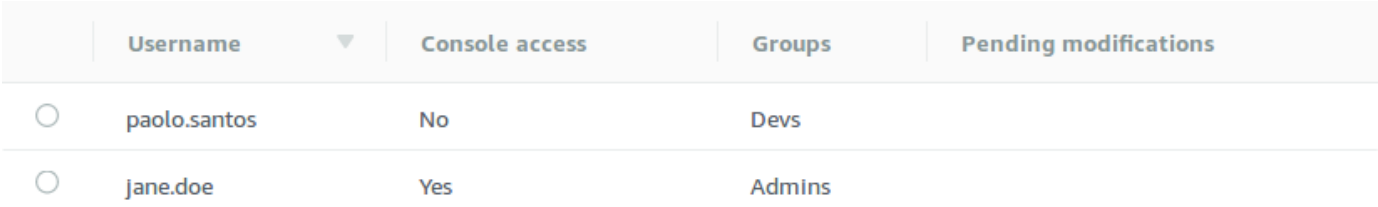
4. (省略可能) 新しい [パスワード] を入力します。
5. (省略可能) ユーザーが属するグループの名前をコンマで区切って追加または削除します (例: `Managers, Admins`)。
6. (省略可能) ユーザーが ActiveMQ ウェブコンソールにアクセスできるようにするには、[ActiveMQ ウェブコンソール] を選択します。
7. ユーザーに対する変更を保存するには、[完了] を選択します。

> **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。詳細については、「Amazon MQ ブローカー設定のライフサイクル」を参照してください。

既存のユーザーを削除するには

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 (例: MyBroker) を選択し、詳細の表示 を選択します。

   MyBroker ページの「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一覧表示されます。



3. サインイン認証情報 (など***MyUser***) を選択し、「 の削除」を選択します。
4. ユーザーの削除を確認するには、Delete ***MyUser***? ダイアログボックスで、Delete 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。詳細については、「Amazon MQ ブローカー設定のライフサイクル」を参照してください。

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。

トピック

- Amazon MQ への接続
- 効果的な Amazon MQ パフォーマンスの確保
- 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

Amazon MQ への接続

以下の設計パターンは、Amazon MQ ブローカーへのアプリケーションの接続の有効性を向上させることができます。

トピック

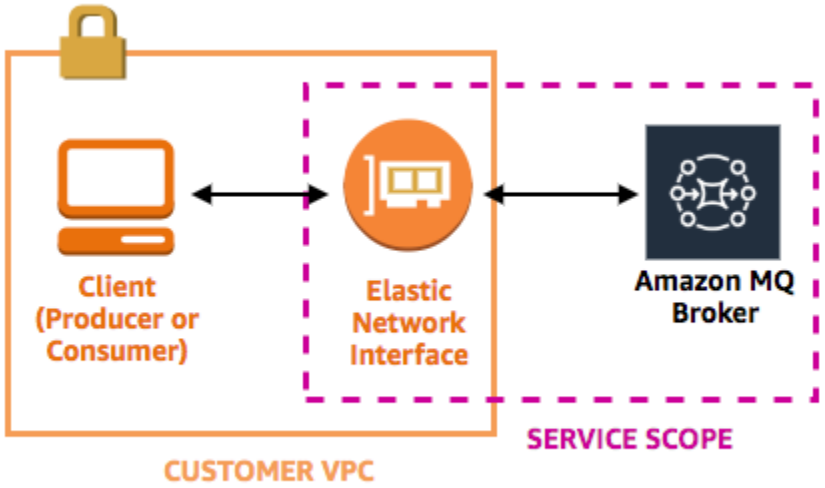
- Amazon MQ Elastic Network Interface を変更または削除しない
- 常に接続プールを使用する
- 常にフェイルオーバートランスポートを使用して複数のブローカーエンドポイントに接続する
- メッセージセレクタを使用しない
- 永続サブスクリプションよりも仮想送信先を優先する
- Amazon VPC ピアリングを使用する場合は、CIDR 範囲 10.0.0.0/16 内のクライアント IP を避けてください。

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 (Getting Started with Amazon MQ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。以下はその例です。

```
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-
east-2.amazonaws.com:61617)?randomize=true
```

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。

効果的な Amazon MQ パフォーマンスの確保

以下の設計パターンは、Amazon MQ ブローカーの有効性とパフォーマンスを向上させることができます。

トピック

- 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
- 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
- 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

- ブローカーのネットワークを正しく設定する

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 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。

このセクションでは、Amazon MQ for ActiveMQ でクロスリージョンデータレプリケーションを設定する方法に関するチュートリアルを提供します。

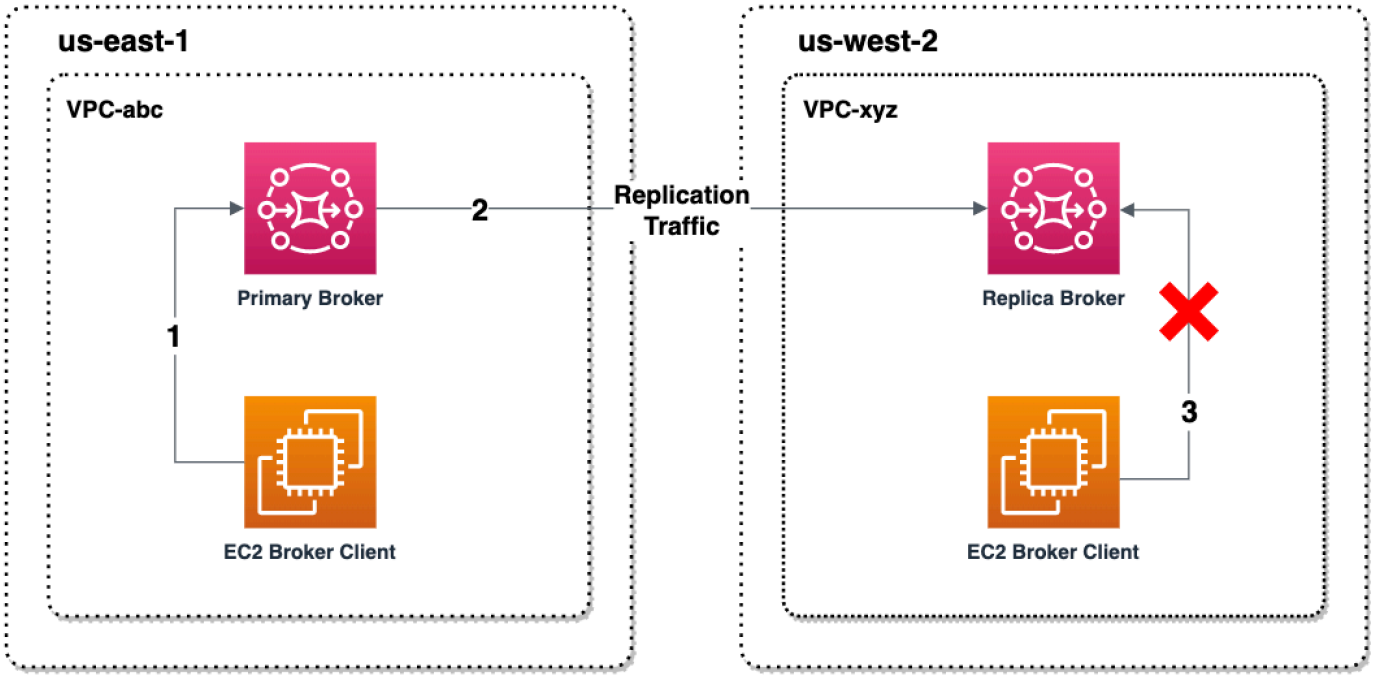
トピック

- Amazon MQ のプライマリブローカーとレプリカブローカー
- 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と削除
- 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
- 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 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、「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と削除」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。

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と削除

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、「Amazon MQ 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
- CRDR ブローカーを削除する

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、「Broker Architecture」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、「ActiveMQ ブローカーの作成と接続」を参照してください。

CRDR ブローカーを削除する

プライマリ 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「unpair」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。

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

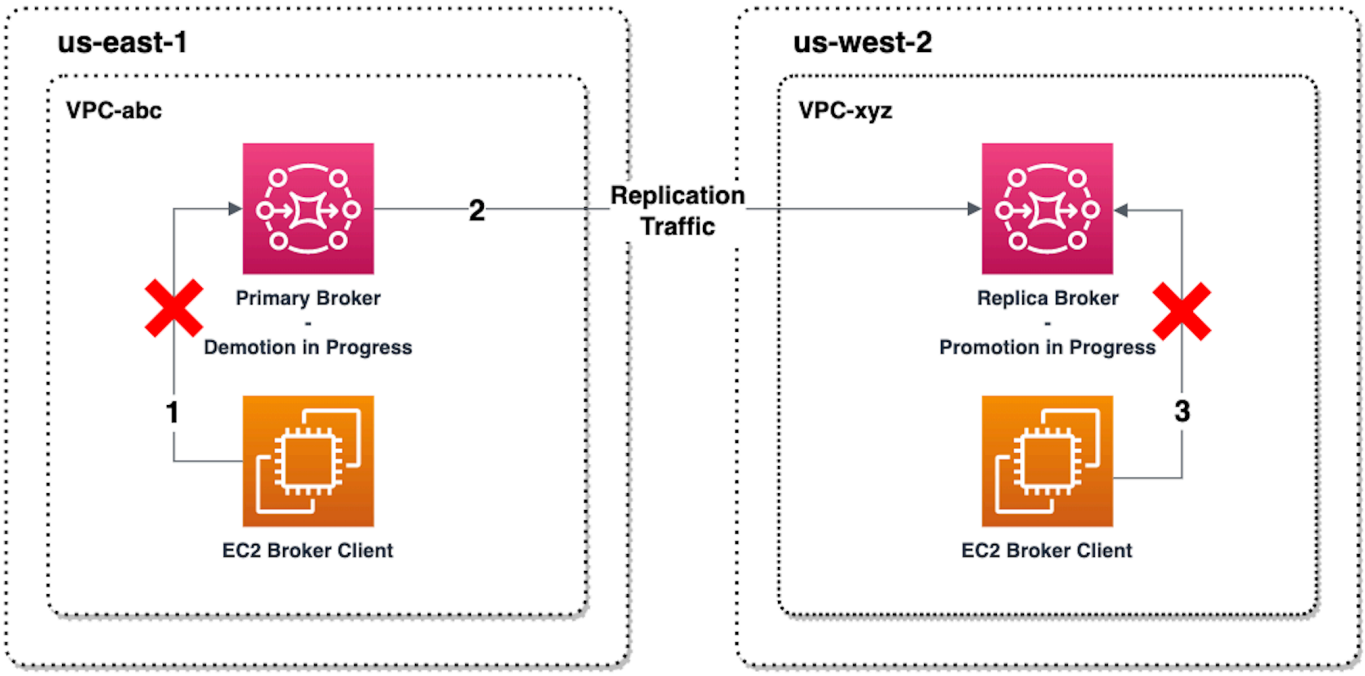
レプリカブローカーをプライマリブローカーのロールに昇格させる場合は、スイッチオーバーまたはフェイルオーバーを開始できます。レプリカブローカーを昇格させると、プライマリブローカーはレプリカブローカーのロールに降格されます。

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。

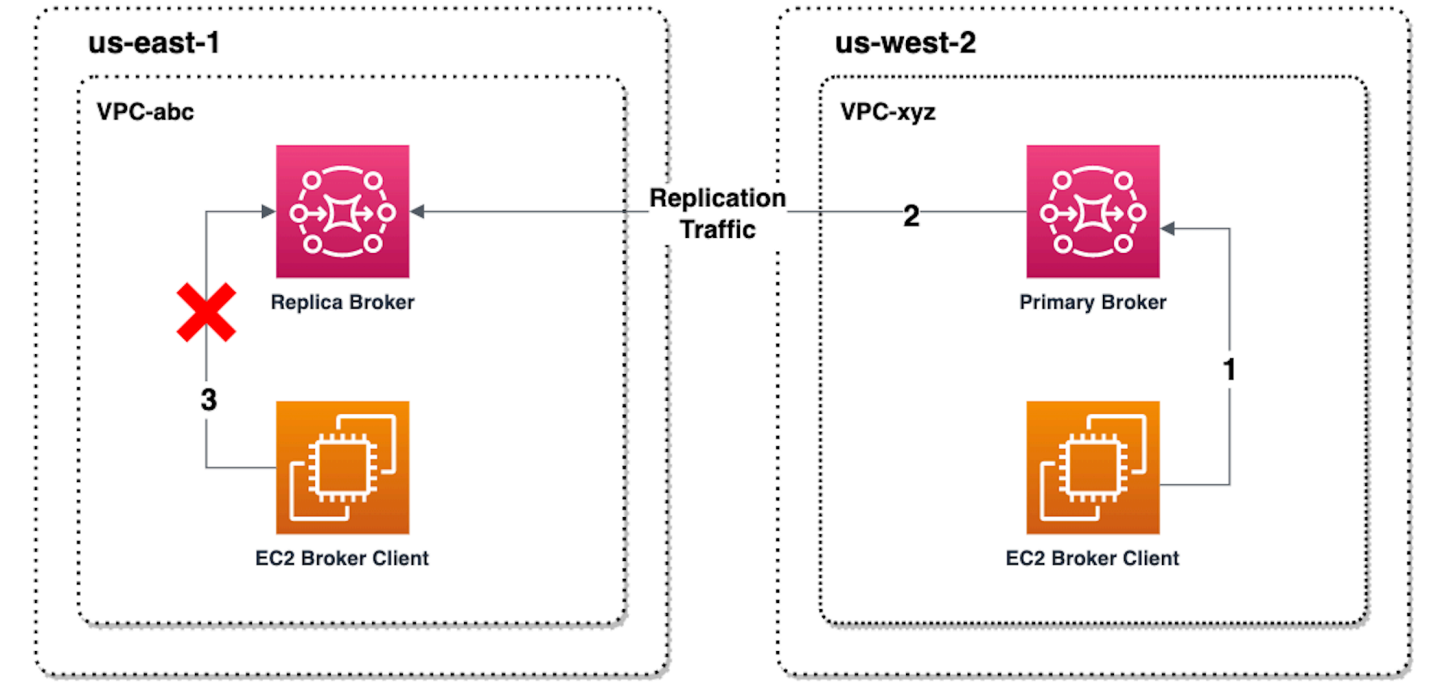
フェイルオーバーでは、一貫性よりも可用性を優先します。このオペレーションが完了すると、ブローカーが同じ状態であるとは限りません。フェイルオーバーの場合、レプリケーションデータが同期されるまで、またはプライマリがシャットダウン信号を受信するまで待つことなく、レプリカブローカーがすぐにクライアントトラフィックの処理に使用可能になることが保証されます。フェイルオーバーが成功するかどうかは、元のプライマリリージョンの正常性にも、リージョン間ネットワークの成功にも依存しません。

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、プライマリブローカーの VPC 内のクライアントは、オペレーションの進行中、およびプライマリブローカーがレプリカに降格している間、さらに状態の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] を選択します。

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。

Amazon MQ for ActiveMQ のクォータ

このトピックでは、Amazon MQ 内のクォータを一覧表示します。以下のクォータの多くは、特定の AWS アカウントで変更できます。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 での現在の接続制限の表示の詳細については CloudWatch、「Amazon を使用した Amazon MQ ブローカーのモニタリング CloudWatch」を参照してください。

> **Note**
>
> Amazon MQ for RabbitMQ のクォータについては、「Amazon MQ for RabbitMQ のクォータ」を参照してください。

トピック

- ブローカー
- Configurations
- [ユーザー]
- データストレージ
- API スロットリング

ブローカー

次の表は、Amazon MQ for ActiveMQ の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	• ブローカーリージョンと AWS アカウントで一意である必要があります。 • 1 ~ 50 文字にする必要があります。

制限	説明
	• 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
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	`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	`mq.*.*large` インスタンスタイプのブローカーに対して 2,000 個。
ネットワークコネクターの数	20
ブローカーあたりのセキュリティグループ	5
でモニタリングされる ActiveMQ の送信先 (キューとトピック) CloudWatch	CloudWatch は、最初の 1000 個の送信先のみをモニタリングします。
でモニタリングされる RabbitMQ 送信先 (キュー) CloudWatch	CloudWatch は、コンシューマーの数順に並べられた最初の 500 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

Configurations

次の表は、Amazon MQ for ActiveMQ 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。

制限	説明
	• 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

[ユーザー]

次の表は、Amazon MQ for ActiveMQ のブローカー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。 • 少なくとも 4 個の一意文字を含める必要があります。 •

制限	説明
	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

次の表は、Amazon MQ for ActiveMQ 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.*.*large` 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	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	`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	`mq.*.*large` 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、サービス帯域幅を維持するために、すべての Amazon MQ APIs にわたって 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 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。ブローカーは、AWS Management Console、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。

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。

トピック

- RabbitMQ エンジン
- RabbitMQ のチュートリアル
- Amazon MQ for RabbitMQ のベストプラクティス
- Amazon MQ for RabbitMQ のクォータ

RabbitMQ エンジン

このセクションでは、RabbitMQ ブローカーとサポートされるプラグインの基本的要素、および Amazon MQ における RabbitMQ ブローカーのアーキテクチャオプションの概要を説明します。

トピック

- 基本的要素
- ブローカーのアーキテクチャ
- Amazon MQ for RabbitMQ ブローカーの設定
- Amazon MQ for RabbitMQ エンジンバージョンの管理

基本的要素

このセクションでは、RabbitMQ on Amazon MQ を理解するうえで不可欠な主要概念を説明します。

トピック

- ブローカー
- ブローカーのデフォルト
- ブローカーインスタンスタイプ
- Amazon MQ for RabbitMQ のサイズ設定ガイドライン
- Configurations
- ユーザー
- プラグイン
- ポリシー

ブローカー

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`、`t3`) およびサイズ (`large`、`micro`) を組み合わせた説明がブローカーインスタンスタイプ (`mq.m5.large` など) になります。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、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。
- 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、「ブローカーのアーキテクチャ」を参照してください。

マイナーバージョンの自動アップグレードを有効にして、RabbitMQ エンジンの新しいマイナーバージョンがリリースされたときに、ブローカーエンジンを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。

サポートされるプロトコル

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)

リスナーポート

Amazon MQ マネージド RabbitMQ ブローカーは、amqps 経由でのアプリケーションレベルの接続、および RabbitMQ ウェブコンソールと Management API を使用したクライアント接続に対して以下のリスナーポートをサポートします。

- リスナーポート 5671 – セキュアな AMQP URL 経由で行われる接続に使用されます。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 amqp 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 および 15671 – RabbitMQ ウェブコンソールまたは Management API 経由でのブローカーへのアクセスには、両方のリスナーポートを区別なく使用できます。

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 です。
- ID。例えば b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9 です。
- 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 です。
- 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 です。

  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 です。

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 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

ブローカーのデフォルト

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、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。

- 単一インスタンス – `AWS-DEFAULT-POLICY-SINGLE-INSTANCE`
- クラスターデプロイ – `AWS-DEFAULT-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。
> 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。

- **`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

- **`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

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
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
	クラスター	8,000,000	4,000	15,000
m5.xlarge	単一インスタンス	該当なし	30,000	8,000
	クラスター	9,000,000	5,000	20,000
m5.2xlarge	単一インスタンス	該当なし	60,000	15,000
	クラスター	10,000,000	6,000	40,000
m5.4xlarge	単一インスタンス	該当なし	150,000	30,000
	クラスター	12,000,000	10,000	100,000

ブローカーインスタンスタイプ

> **Important**
>
> 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。

インスタンスタイプ	vCPU	メモリ (GiB)	ネットワークパフォーマンス	ユースケース
mq.t3.micro	2	1	低	評価

インスタンスタイプ	vCPU	メモリ (GiB)	ネットワークパフォーマンス	ユースケース
				Important `mq.t3.micro` インスタンスタイプはクラスターデプロイをサポートしません。
`mq.m5.large`	2	8	高い	本番稼働
`mq.m5.xlarge`	4	16	高い	本番稼働
`mq.m5.2xlarge`	8	32	高い	
`mq.m5.4xlarge`	16	64	高い	

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
- 送信されたメッセージの量
- 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は、アプリケーションのパフォーマンスのテストに使用できます。ブローカーインスタンスタイプが大きいほど、クライアントとキューの本番稼働レベル、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。

ブローカーをテストして、ワークロード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。以下のサイジングガイドラインを使用して、アプリケーションに最適なインスタンスタイプを決定します。

インスタンスタイプ	デプロイモード	接続の最大数	最大チャネル数
t3.micro	単一インスタンス	500	1,500
m5.large	単一インスタンス	5,000	15,000
	クラスター	15,000	45,000
m5.xlarge	単一インスタンス	10,000	30,000
	クラスター	30,000	90,000
m5.2xlarge	単一インスタンス	20,000	60,000
	クラスター	60,000	180,000
m5.4xlarge	単一インスタンス	40,000	120,000
	クラスター	120,000	360,000

接続またはチャンネルの制限を超えると、次のエラーメッセージが返されます。

チャンネル

```
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 (1500)"
```

Connection

```
ConnectionClosedByBroker 500 "NOT_ALLOWED - connection refused: node connection limit (500) is reached"
```

Configurations

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。

> **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。詳細については、「Amazon MQ ブローカー設定のライフサイクル」を参照してください。
> 現在、設定を削除することはできません。

設定を作成、編集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。

- Creating and applying broker configurations
- RabbitMQ Broker Configurations

設定に対して行った変更を追跡するために、設定リビジョンを作成できます。詳細については、「Creating and applying broker configurations」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

ユーザー

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、認証される必要があります。各クライアント接続は仮想ホスト (vhost) もターゲットにしており、ユーザーにはこのホストに対する一連の許可が必要です。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。ユーザーの認証情報、およびターゲット vhost は、接続の確立時に指定されます。

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 はユーザー名「ゲスト」をサポートしておらず、新しいブローカーを作成するとデフォルトのゲストアカウントが削除されます。Amazon MQ は、お客様が作成した「ゲスト」というアカウントも定期的に削除します。

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 ログを含む他の 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。
- ブローカーの作成時に設定した管理者パスワードを忘れた場合は、認証情報をリセットできません。複数の管理者を作成した場合は、別の管理者ユーザーを使用してログインし、認証情報をリセットまたは再作成できます。管理者ユーザーが 1 人しかない場合は、ブローカーを削除し、新しい認証情報で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前に、メッセージを使用またはバックアップすることをお勧めします。

`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。

プラグイン

Amazon MQ for RabbitMQ は、この Management API と RabbitMQ ウェブコンソールを動作させる RabbitMQ の Management プラグインをサポートします。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。

管理プラグインに加えて、Amazon MQ for RabbitMQ は以下のプラグインもサポートします。

トピック

- シャベルプラグイン
- フェデレーションプラグイン
- コンシステントハッシュエクスチェンジプラグイン

シャベルプラグイン

Amazon MQ マネージドブローカーは RabbitMQ シャベルをサポートしており、1 つのブローカーインスタンス上にあるキューとエクスチェンジからのメッセージを、別のブローカーインスタンスに移動することを可能にします。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。

Amazon MQ マネージド RabbitMQ ブローカーは、動的シャベルをサポートします。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プログラム的にいつでも開始および停止できます。例えば、RabbitMQ Management API を使用して、以下の API エンドポイントに対する PUT リクエストを作成し、動的シャベルを設定することができます。この例では、{vhost} をブローカーの vhost の名前、{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で動的シャベルを設定します。同様に、src-exchange および 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することもできます。

```
{
                "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-
west-2.amazonaws.com:5671",
                    "dest-queue": "destination-queue-name"
                }
                }
```

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。シャベルを構成できるのは、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、プライベートブローカーのソースとパブリックブローカーの宛先の間のみです。

動的シャベルの使用の詳細については、「RabbitMQ dynamic shovel plugin」を参照してください。

Note

Amazon MQ は、静的シャベルの使用をサポートしません。

フェデレーションプラグイン

Amazon MQ は、フェデレートされた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。フェデレーティッドキューとエクスチェンジは point-to-point 、リンクを使用して他のブローカーのピアに接続します。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。Management API を使用して、以下のリクエストボディで `/api/parameters/federation-upstream/%2f/my-upstream` に対する `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。Management API を使用して、以下のリクエストボディで `/api/policies/%2f/federate-me` に対する `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\.` の使用は、名前が「amq」で始まるすべてのエクスチェンジに対してフェデレーションが有効化されることを確実にします。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。

フェデレーションプラグインの設定に関する詳細については、「RabbitMQ federation plugin」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

デフォルトで、Amazon MQ for RabbitMQ はコンシステントハッシュエクスチェンジタイプのプラグインをサポートします。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交換にバインド number-as-a-string 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ディングウェイトを決定する 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。

コンシステントハッシュ交換の詳細については、 GitHub ウェブサイトのRabbitMQ コンシステントハッシュ交換タイプ」を参照してください。

ポリシー

Amazon MQ が推奨するデフォルト値を使用して、カスタム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**
>
> 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、「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`***。 を、 `mq.m5.large`クラスター**`8000000`**の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て Amazon Amazon MQ の推奨値*`number-of-messages`*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 [Virtual host limits] (仮想ホスト制限) リストにが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。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て Amazon MQ 推奨値*`number-of-messages`*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
```

ブローカーのアーキテクチャ

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

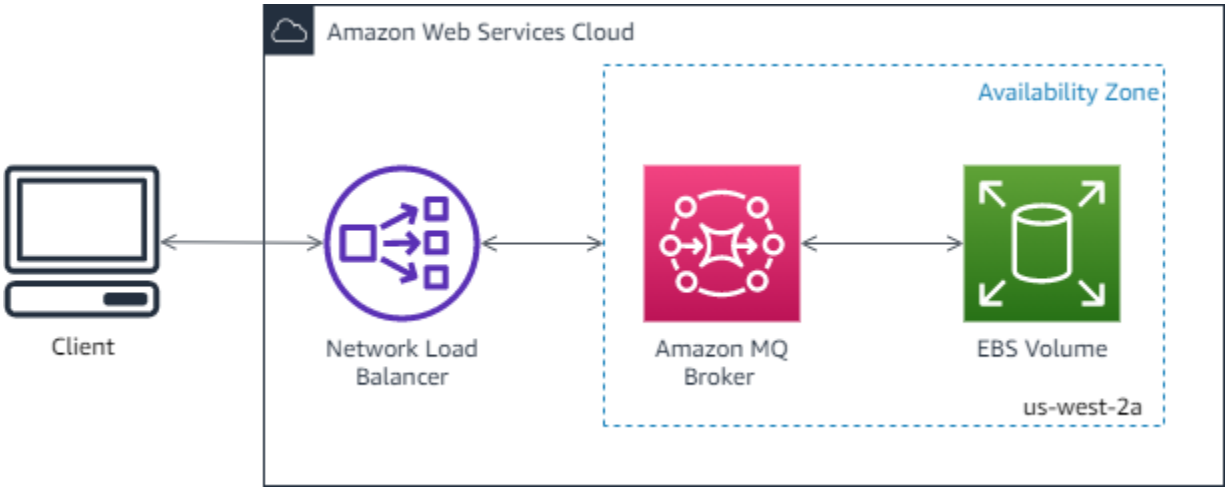
- 単一インスタンスブローカー
- 高可用性対応のクラスターデプロイ

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。



高可用性対応の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**
>
> 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 “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

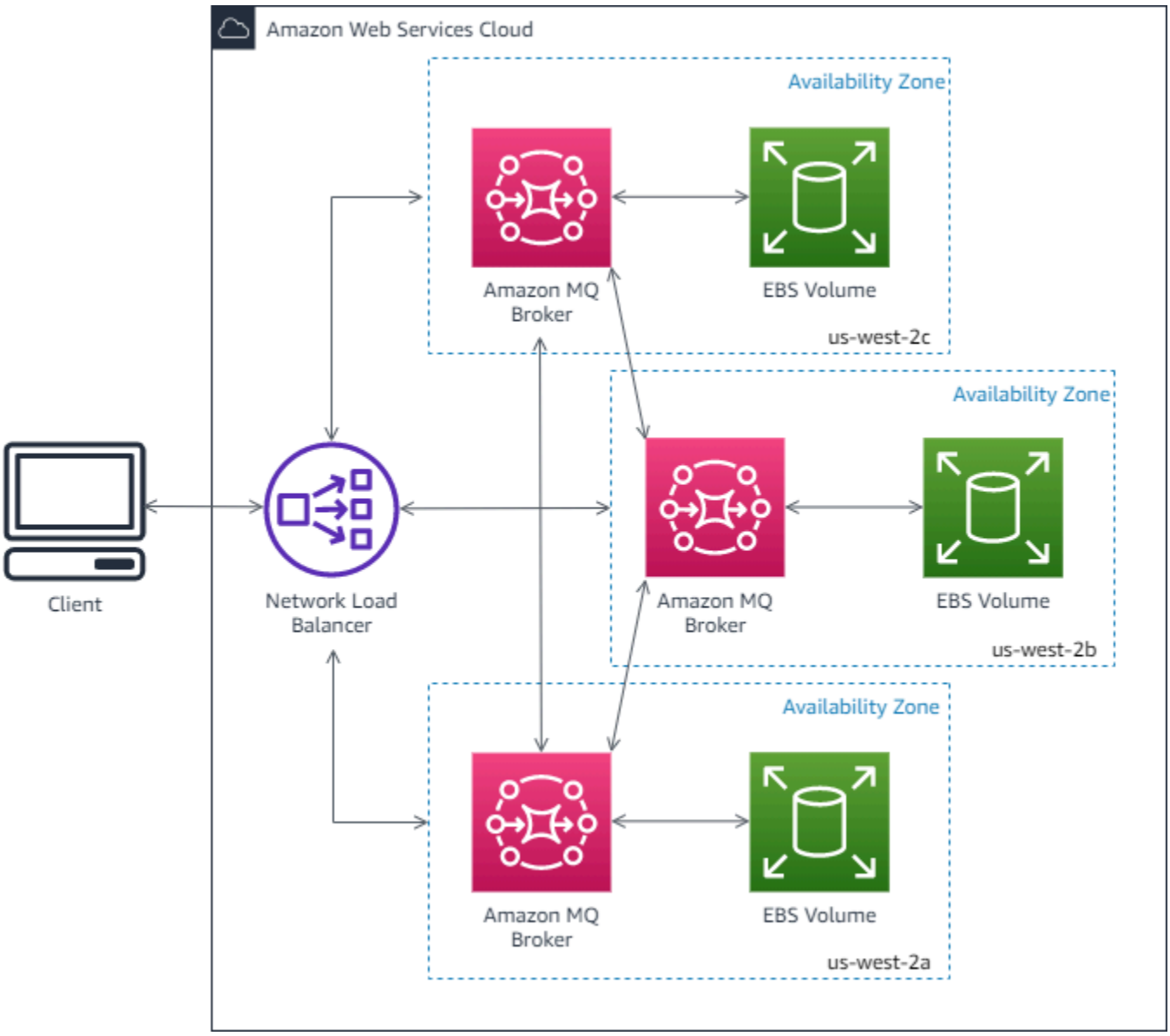
の他すべてのキュー操作をブロックします。メンテナンスウィンドウ中におけるキューの同期の影響は、キューを短くしておくことによって軽減できます。

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すると、Amazon MQ2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。

Important

Amazon MQ はクォーラムキューをサポートしません。クォーラムキュー機能フラグの有効化とクォーラムキューの作成は、データ損失の原因になります。

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 ブローカーの設定

configuration には、RabbitMQ 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。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。

トピック

- RabbitMQ ブローカー設定の作成、編集、適用
- RabbitMQ 設定ポリシー

RabbitMQ ブローカー設定の作成、編集、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。

詳細については、次を参照してください。

- Configurations
- Amazon MQ ブローカー設定のライフサイクル

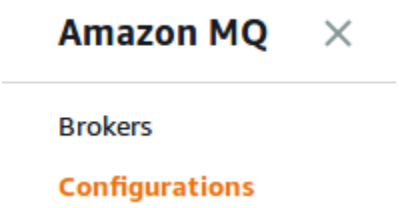
以下の例では、 AWS Management Consoleを使用して Amazon MQ ブローカーの設定を作成および適用する方法を示します。

トピック

- 新しい設定の作成
- 新しい設定リビジョンの作成
- 設定リビジョンをブローカーに適用する
- 設定のリビジョンの編集

新しい設定の作成

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. [Create configuration] (設定の作成) ページの [Details] (詳細) セクションで [Configuration name] (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. [Create configuration] (設定を作成) をクリックします。

新しい設定リビジョンの作成

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

> **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 ページには、新しい設定リビジョンが使用するブローカーエンジンのタイプとバージョン (RabbitMQ 3.xx.xx など) 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. [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。

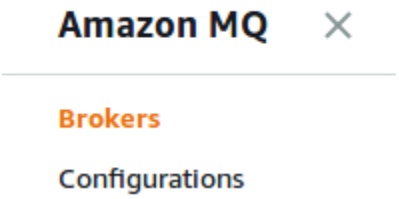
5. (オプション) `A description of the changes in this revision` を入力します。
6. [保存] を選択します。

設定の新しいリビジョンが保存されます。

> **Important**
>
> 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。詳細については、「Amazon MQ ブローカー設定のライフサイクル」を参照してください。
> 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

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 (などMyBroker) を選択し、編集 を選択します。
3. 「編集***MyBroker***」ページの「設定」セクションで、「設定」と「リビジョン」を選択し、「スケジュールの変更」を選択します。
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、ブローカーはオフラインになります。

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。

設定のリビジョンの編集

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 (などMyBroker) を選択し、編集 を選択します。
3. ***MyBroker*** ページで、編集 を選択します。
4. 「編集***MyBroker***」ページの「設定」セクションで、「設定」と「リビジョン」を選択し、「編集」を選択します。

> **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 ページには、設定が使用するブローカーエンジンの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. [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**
>
> 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。詳細については、「Amazon MQ ブローカー設定のライフサイクル」を参照してください。
> 現在、設定を削除することはできません。

RabbitMQ 設定ポリシー

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。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 Management Console または Management API を使用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。ブローカーに設定を追加する方法の詳細については、「Creating and applying broker configurations」を参照してください。

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示し、 はパッチバージョン番号Z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更された場合、バージョン変更はマイナーと見なされます。例えば、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の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のバージョンのアップグレード」を参照してください。

> **Important**
>
> Amazon MQ はクォーラムキューまたはストリームをサポートしません。これらの機能フラグの有効化とクォーラムキューまたはストリームの作成は、データ損失の原因になります。Amazon MQ 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。

Amazon MQ for RabbitMQ でサポートされてい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には、ブローカーエンジンバージョンがサポート終了になる時期が表示されます。バージョンがサポート終了になると、Amazon MQ は、このバージョンのすべてのブローカーを次にサポートされているバージョンに自動的にアップグレードします。Amazon MQ は、バージョンがサポートを終了する少なくとも 90 日前に通知します。

RabbitMQ バージョン	Amazon MQ のサポート終了
3.12 (推奨)	
3.11	
3.10	2024 年 10 月 15 日
3.9	2024 年 9 月 16 日
3.8	2024 年 8 月 15 日

新しい Amazon MQ for RabbitMQ 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 RabbitMQ エンジンバージョンを指定できます。を使用してブローカー AWS Management Console を作成する場合、Amazon MQ は自動的に最新のエンジンバージョン番号にデフォルト設定されます。 AWS CLI または Amazon MQ API を使用してブローカーを作成する場合は、エンジンのバージョン番号が必要です。バージョン番号を指定しない場合は、操作で例外が発生します。詳細については、AWS CLI コマンドリファレンスの「`create-broker`」、および Amazon MQ REST API リファレンスの「`CreateBroker`」を参照してください。

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、次にサポートされているメジャー、マイナー、またはパッチ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 中にブローカーをサポートされている最新のパッチバージョンにアップグレードします。 Amazon MQ

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

> **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。更新時にマイナーバージョンをスキップすることはできません (例: 3.8.x から 3.11.x)。

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、ミラーリングされたキューは再起動時に同期する必要があります。キューが長いほど、キュー同期プロセスに時間がかかる場合があります。キュー同期プロセス中は、コンシューマーとプロデューサーはキューを使用できません。キュー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、トラフィックが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、「」を参照してくださいAmazon MQ for RabbitMQ のベストプラクティス。

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 `RABBITMQ`、*`instance-type`* を `mq.m5.large` に置き換えます。

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、Node.js、Python、.NET などのさまざまなプログラミング言語の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 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
- Amazon MQ for RabbitMQ でPython Pika を使う
- RabbitMQ の一時停止されたキュー同期の解決

ブローカー設定の編集

AWS Management Consoleを使用して、CloudWatch Logs の有効化または無効化などのブローカー設定を編集することができます。

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。

   AWS からの新しいバージョンのリリースに伴ってブローカーをアップグレードするには、[Enable automatic minor version upgrades] (自動マイナーバージョンアップグレードの有効化) 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、およびキューからメッセージを受信するようにコンシューマを設定する手順を説明します。

トピック

- 前提条件
- 許可
- ステップ 1: 基本的な Python Pika クライアントを作成する
- 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
- ステップ 3: コンシューマを作成してメッセージを受信する
- 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
- 次のステップ

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
```

許可

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。以下の表は、正規表現 (regexp) パターンとして必要な最低限の許可を説明しています。

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、「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 です。

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: コンシューマを作成してメッセージを受信する

キューから 1 つのメッセージを受信するコンシューマを作成するには、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、BasicMessageReciever は BasicPikaClient から継承し、単一のメッセージを受信し、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 BasicMessageReciever 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
- 次のステップ
- 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、「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。キューが、[Features] (特徴) 列に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、[Restart sync] (同期を再開) をクリックします。

> **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。

次のステップ

- 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 ブローカーのロギングとモニタリング 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、「Amazon MQ for RabbitMQ インスタンスタイプ」および「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、「the section called “ブローカーのデフォルト”」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。

Amazon MQ for RabbitMQ のベストプラクティス

このセクションは、Amazon MQ での RabbitMQ 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。

Important

Amazon MQ はクォーラムキューをサポートしません。クォーラムキュー機能フラグの有効化とクォーラムキューの作成は、データ損失の原因になります。

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。

Important

Amazon MQ for RabbitMQ はユーザー名「ゲスト」をサポートしておらず、新しいブローカーを作成するとデフォルトのゲストアカウントが削除されます。Amazon MQ は、お客様が作成した「ゲスト」というアカウントも定期的に削除します。

トピック

- レイジーキューを有効にする
- 永続キューと持続キューを使用する
- キューを短くしておく
- 承認と確認を設定する
- プリフェッチを設定する
- Celery を設定
- ネットワーク障害から自動的に回復する
- RabbitMQ ブローカーの Classic Queue v2 を有効にする

レイジーキューを有効にする

大量のメッセージを処理する非常に長いキューを使用している場合、レイジーキューを有効にするとブローカーのパフォーマンスが向上します。

RabbitMQ のデフォルト動作では、メッセージをメモリにキャッシュし、ブローカーでより多くの使用可能なメモリが必要となった場合にのみ、それらをディスクに移動します。メモリからディスクへのメッセージの移動には時間がかかり、メッセージ処理が停止します。レイジーキューは、メッセージをできるだけ早くディスクに保存することで、メモリからディスクへのプロセスを大幅に高速化し、メモリにキャッシュされるメッセージを減らします。

レイジーキューは、宣言時に `queue.declare` 引数を設定する、または RabbitMQ のマネジメントコンソールでポリシーを設定することによって有効にできます。以下の例は、RabbitMQ Java クライアントライブラリを使用したレイジーキューの宣言を示しています。

```
Map<String, Object> args = new HashMap<String, Object>();
args.put("x-queue-mode", "lazy");
channel.queueDeclare("myqueue", false, false, false, args);
```

3.12.13 以降のすべての Amazon MQ for RabbitMQ キューは、デフォルトでレイジーキューとして動作します。Amazon MQ for RabbitMQ の最新バージョンにアップグレードするには、「」を参照してください???。

> **Note**
>
> レイジーキューを有効にすると、ディスク I/O 操作が増加する場合があります。

永続キュー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損失の防止に役立ちます。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。

```
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());
```

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。で RabbitMemUsedとRabbitMqMemLimitブローカーノードのメトリクス CloudWatchを比較することで、メモリ使用量を確認できます。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。

> **Warning**
>
> 多くのリソースを使用して実行されているブローカーは再起動しないでください。
> キューの同期が一時停止しているときにブローカーを再起動すると、ブローカーは同期プロセスを再開します。これにより、メッセージがストレージからノードメモリに転送されるため、ブローカーリソースがさらに低下し、その結果、ブローカーが REBOOT_IN_PROGRESS 状態のまま応答しなくなる可能性があります。

承認と確認を設定する

クライアントアプリケーションによるメッセージの配信確認と消費確認のブローカーへの返送は、コンシューマー承認として知られています。同様に、パブリッシャーに確認を送信するプロセスはパブリッシャー確認として知られています。RabbitMQ ブローカーの使用時におけるデータの安全性を確実にするには、承認と確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 を使用する場合、承認は basic.consume を設定して有効にする、または basic.code メソッドを使用してメッセージを取得するときに有効にすることができます。

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。

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。

Celery を設定

Python Celery は、有用な情報の検索と処理をより困難にする多くの不要なメッセージを送信します。ノイズを減らして処理を容易にするには、次のコマンドを入力します。

```
celery -A app_name worker --without-heartbeat --without-gossip --without-mingle
```

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。

RabbitMQ ブローカーの Classic Queue v2 を有効にする

ブローカーエンジンバージョン 3.10 および 3.11 で Classic Queue v2 (CQv2) を有効にして、次のようなパフォーマンスを向上させることをお勧めします。

- メモリ使用量を減らす
- コンシューマーへの配信を改善する
- コンシューマーがプロデューサーに遅れずに対応するワークロードのスループットを向上させる

3.12.13 以降のすべての Amazon MQ for RabbitMQ キューは、デフォルトで CQv2 を使用します。Amazon MQ for RabbitMQ の最新バージョンにアップグレードするには、「」を参照してください???。

CQv1 から CQv2 への移行

CQv2 を使用するには、まず`classic_mirrored_queue_version`機能フラグを有効にする必要があります。機能フラグの詳細については、「機能フラグを有効にする方法」を参照してください。

CQv1 から CQv2 に移行するには、新しいキューポリシーを作成するか、ポリシーキー定義を に設定して既存のキュー`queue-version`ポリシーを編集する必要があります2。ポリシーの適用の詳細については、「」を参照してくださいポリシー。キューポリシーで CQv2 を有効にする方法の詳細については、RabbitMQ ドキュメントの「Classic Queues」を参照してください。

移行を開始する前に、他のパフォーマンスに関するベストプラクティスに従うことをおすすめします。

キューポリシーを使用している場合は、キューポリシーを削除すると CQv2 キューが CQv1 にダウングレードされます。CQv2 キューを CQv1 にダウングレードすることはお勧めしません。RabbitMQ はキューのディスク上の表現を変換するためです。キューの深さが深い場合、これはメモリを大量に消費し、時間がかかる可能性があります。

Amazon MQ for RabbitMQ のクォータ

このトピックでは、Amazon MQ 内のクォータを一覧表示します。以下のクォータの多くは、特定の AWS アカウントに対して変更することが可能です。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

- データストレージ
- API スロットリング

ブローカー

次の表は、Amazon MQ for RabbitMQ の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	• ブローカーのリージョンと AWS アカウントで一意である必要があります。 • 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
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

データストレージ

次の表は、Amazon MQ for RabbitMQ 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.*.*large` 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。

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 のセキュリティ

AWS では、クラウドのセキュリティが最優先事項です。AWS のお客様は、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利点を得られます。

セキュリティは、AWS とお客様の間の共有責任です。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 は、AWS クラウドで AWS のサービスを実行するインフラストラクチャを保護する責任を負います。また、AWS は、使用するサービスを安全に提供します。AWS コンプライアンスプログラムの一環として、サードパーティーの監査が定期的にセキュリティの有効性をテストおよび検証しています。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による AWS 対象範囲内のサービスを参照してください。
- クラウド内のセキュリティ – お客様の責任は、使用する AWS のサービスに応じて判断されます。また、お客様は、お客様のデータの機密性、企業の要件、および適用可能な法律および規制などの他の要因についても責任を担います。

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とセキュア化に役立つ AWS のその他のサービスを使用する方法も学びます。

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

Amazon MQ でのデータ保護には、AWS の責任共有モデルが適用されます。このモデルで説明されているように、AWS は、AWS クラウド のすべて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を担います。ユーザーには、このインフラストラクチャでホストされてい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にも責任があります。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の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。

データを保護するため、AWS アカウント の認証情報を保護し、AWS IAM Identity Center または AWS Identity and Access Management (IAM) を使用して個々のユーザーをセットアップすることをお勧めします。こうすると、それぞれのジョブを遂行するために必要なアクセス許可のみを各ユーザーに付与できます。また、次の方法でデータを保護することをお勧めします。

- 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 が必須です。TLS 1.3 が推奨されます。
- AWS CloudTrail で API とユーザーアクティビティログをセットアップします。
- AWS のサービス内でデフォルトである、すべてのセキュリティ管理に加え、AWS の暗号化ソリューションを使用します。
- Amazon Macie などの高度なマネージド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 を使用して AWS にアクセスするときに FIPS 140-2 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) 140-2」を参照してください。

顧客の E メールアドレスなどの機密情報や重要情報は、タグや Name フィールドなどの自由形式のフィールドに入力しないことを強くお勧めします。これには、コンソール、API、AWS CLI、または AWS SDK を使用して Amazon MQ またはその他の AWS のサービス で作業する場合が含まれます。名前に使用する自由記述のテキストフィールドやタグに入力したデータは、課金や診断ログに使用される場合があります。外部サーバーへ URL を提供する場合は、そのサーバーへのリクエストを検証するための認証情報を URL に含めないように強くお勧めします。

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 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。

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、透過的なサーバー側暗号化を提供するために AWS Key Management Service (KMS) と統合します。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は、保管中のデータの暗号化に Amazon MQ が使用する AWS KMS key を指定できます。KMS キーを指定しない場合、Amazon MQ は自動的に AWS 所有の KMS キーを作成し、ユーザーに代わってそれを使用します。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 がユーザーに代わって作成、管理、および使用する、ユーザーのアカウントにある 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。アクセス権を取り消す必要がある場合は、ブローカーを削除することをお勧めします。
> - 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、アカウントで KMS キーを使用する許可を Amazon EFS に提供する付与を取り消すと、すぐには有効にはなりません。
> - 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。
> - キーを無効にするか、付与を取り消しても、すぐには有効にはなりません。

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: クラスターのデプロイ

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 を通過するだけで、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層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するために役立つ AWS のサービスです。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、追加費用なしで使用できる AWS のサービスです。

トピック

- 対象者
- アイデンティティを使用した認証
- ポリシーを使用したアクセスの管理
- Amazon MQ で IAM が機能する仕組み
- Amazon MQ のアイデンティティベースポリシーの例
- Amazon MQ の API 認証と認可
- AWS Amazon MQ の マネージドポリシー
- Amazon MQ のサービスリンクロールの使用
- 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

対象者

AWS Identity and Access Management (IAM) の使用方法は、Amazon MQ で行う作業に応じて異なります。

サービスユーザー – 業務を行うために Amazon MQ サービスを使用する場合は、管理者から必要な認証情報と許可が提供されます。業務のために使用する Amazon MQ 機能が増えるにつれて、追加の許可が必要になる可能性があります。アクセスの管理方法を理解しておくことは、管理者に適

切な許可をリクエストするために役に立ちます。Amazon MQ の機能にアクセスできない場合は、「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」を参照してください。

サービス管理者 – 社内の Amazon MQ リソースを担当している場合は、Amazon MQ に対する完全なアクセス権があると思われます。サービスのユーザーがどの Amazon MQ 機能やリソースにアクセスするかを決めるのは管理者の仕事です。その後、IAM 管理者にリクエストを送信して、サービスユーザーのアクセス許可を変更する必要があります。このページの情報を確認して、IAM の基本概念を理解してください。会社で Amazon MQ と IAM を併用する方法の詳細については、「Amazon MQ で IAM が機能する仕組み」を参照してください。

IAM 管理者 – IAM 管理者には、Amazon MQ へのアクセスを管理するポリシーの作成方法の詳細を理解することが推奨されます。IAM で使用できる 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。

アイデンティティを使用した認証

認証とは、アイデンティティ認証情報を使用して AWSにサインインする方法です。ユーザーは、AWS アカウントのルートユーザーもしくは IAM ユーザーとして、または IAM ロールを引き受けることによって、認証を受ける (AWS にサインインする) 必要があります。

ID ソースから提供された認証情報を使用すると、フェデレーティッドアイデンティティとして AWS にサインインできます。AWS IAM Identity Center(IAM Identity Center) ユーザー、会社のシングルサインオン認証、Google や Facebook の認証情報はフェデレーティッドアイデンティティの例です。フェデレーティッドアイデンティティとしてサインインする場合、IAM ロールを使用して、前もって管理者により ID フェデレーションが設定されています。フェデレーションを使用して AWSにアクセスする場合、間接的にロールを引き受けることになります。

ユーザーのタイプに応じて、AWS Management Console または AWSアクセスポータルにサインインできます。AWS へのサインインの詳細については、「AWS サインイン User Guide」の「How to sign in to your AWS アカウント」を参照してください。

プログラムを使用して AWSにアクセスする場合、AWS は Software Development Kit (SDK) とコマンドラインインターフェイス (CLI) を提供し、認証情報を使用してリクエストに暗号で署名します。AWS ツールを使用しない場合は、リクエストに自分で署名する必要があります。リクエストに署名する推奨方法の使用については、IAM ユーザーガイドの「AWS API リクエストの署名」を参照してください。

使用する認証方法を問わず、セキュリティ情報の提供を追加でリクエストされる場合もあります。例えば、AWS では多要素認証 (MFA) を使用してアカウントのセキュリティを高めることを推奨してい

ます。詳細については、「AWS IAM Identity Center User Guide」の「Multi-factor authentication」および「IAM ユーザーガイド」の「AWS での多要素認証 (MFA) の使用」を参照してください。

AWS アカウントのルートユーザー

AWS アカウントを作成する場合は、そのアカウント内のすべての AWS のサービスとリソースに対して完全なアクセス権を持つ 1 つのサインインアイデンティティから始めます。この ID は AWS アカウントのルートユーザーと呼ばれ、アカウントの作成に使用した E メールアドレスとパスワードでサインインすることによってアクセスできます。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。ルートユーザーの認証情報は保護し、ルートユーザーでしか実行できないタスクを実行するときに使用します。ルートユーザーとしてサインインする必要がある全タスクのリストについては、「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許可を持つ AWS アカウント内のアイデンティティです。可能であれば、パスワードやアクセスキーなどの長期的な認証情報を保有する IAM ユーザーを作成する代わりに、一時的な認証情報を使用することをお勧めします。ただし、IAM ユーザーとの長期的な認証情報が必要な特定のユースケースがある場合は、アクセスキーをローテーション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「長期的な認証情報を必要とするユースケースのためにアクセスキーを定期的にローテーションする」を参照してください。

IAM グループは、IAM ユーザーの集団を指定するアイデンティティです。グループとしてサインインすることはできません。グループを使用すると、一度に複数のユーザーにアクセス許可を指定できます。ユーザーの規模が大きい場合、グループを使用することでアクセス許可の管理が容易になります。例えば、IAMAdmins という名前のグループを作成し、そのグループに IAM リソースを管理するアクセス許可を与えることができます。

ユーザーは、ロールとは異なります。ユーザーは 1 人の人または 1 つのアプリケーションに一意に関連付けられますが、ロールはそれを必要とする任意の人が引き受けるようになっています。ユーザーには永続的な長期の認証情報がありますが、ロールでは一時的な認証情報が提供されます。詳細については、「IAM ユーザーガイド」の「IAM ユーザーの作成が適している場合 (ロールではなく)」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、AWS アカウント内のアイデンティティです。これは IAM ユーザーに似ていますが、特定のユーザーには関連付けられていません。ロールを切り替え

ることにより、AWS Management Console で一時的に IAM ロールを引き受けることができます。ロールを引き受けるには、AWS CLI または AWS API オペレーションを呼び出すか、カスタム URL を使用します。ロールを使用する方法の詳細については、「IAM ユーザーガイド」の「IAM ロールを使用する」を参照してください。

IAM ロールと一時的な認証情報は、次のような状況で役立ちます。

- フェデレーションユーザーアクセス - フェデレーティッド ID にアクセス許可を割り当てるには、ロールを作成し、そのロールのアクセス許可を定義します。フェデレーティッド ID が認証されると、その ID はロールに関連付けられ、ロールで定義されているアクセス許可が付与されます。フェデレーションの詳細については、「IAM ユーザーガイド」の「サードパーティーアイデンティティプロバイダー向けロールの作成」を参照してください。IAM アイデンティティセンターを使用する場合は、アクセス許可セットを設定します。アイデンティティが認証後にアクセスできるものを制御するため、IAM アイデンティティセンターは、アクセス許可セットを IAM のロールに関連付けます。アクセス許可セットの詳細については、「AWS IAM Identity Center User Guide」の「Permission sets」を参照してください。
- 一時的な IAM ユーザー許可：IAM ユーザーまたはロールは、特定のタスクに対して複数の異なる許可を一時的に IAM ロールで引き受けることができます。
- クロスアカウントアクセス - IAM ロールを使用して、自分のアカウントのリソースへのアクセスを別のアカウントの人物 (信頼できるプリンシパル) に許可できます。クロスアカウントアクセスを許可する主な方法は、ロールを使用することです。ただし、一部の AWS のサービスでは、(ロールをプロキシとして使用する代わりに) リソースにポリシーを直接アタッチできます。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違いについては、IAM ユーザーガイドのIAM ロールとリソースベースのポリシーとの相違点を参照してください。
- クロスサービスアクセス - 一部の AWS のサービスでは、他の AWS のサービスの機能を使用します。例えば、あるサービスで呼び出しを行うと、通常そのサービスによって Amazon EC2 でアプリケーションが実行されたり、Amazon S3 にオブジェクトが保存されたりします。サービスでは、呼び出し元プリンシパルの許可、サービスロール、またはサービスリンクロールを使用してこれを行う場合があります。
  - 転送アクセスセッション (FAS) – IAM ユーザーまたはロールを使用して AWS でアクションを実行するユーザーは、プリンシパルと見なされます。一部のサービスを使用する際に、あるアクションを実行すると別のサービスの別のアクションが開始されることがあります。FAS は、AWS のサービスを呼び出すプリンシパルのアクセス許可を、リクエスト元の AWS のサービスと組み合わせて使用し、ダウンストリームサービスに対してリクエストを行います。FAS リクエストが行われるのは、他の AWS のサービスやリソースとのやり取りを完了する必要があるリクエストをサービスが受信した場合のみです。この場合、両方のアクションを実行するた

めの許可が必要です。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。

- サービスロール - サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、「IAM ユーザーガイド」の「AWS のサービスにアクセス許可を委任するロールの作成」を参照してください。
- サービスにリンクされたロール - サービスにリンクされたロールは、AWS のサービスにリンクされたサービスロールの一種です。サービスがロールを引き受け、ユーザーに代わってアクションを実行できるようになります。サービスにリンクされたロールは、AWS アカウント内に表示され、サービスによって所有されます。IAM 管理者は、サービスにリンクされたロールの許可を表示できますが、編集はできません。

- Amazon EC2 で実行されているアプリケーション：EC2 インスタンスで実行され、AWS CLI または AWS API 要求を行っているアプリケーションの一時的な認証情報を管理するには、IAM ロールを使用できます。これは、EC2 インスタンス内でのアクセスキーの保存に推奨されます。AWS ロールを EC2 インスタンスに割り当て、そのすべてのアプリケーションで使用できるようにするには、インスタンスに添付されたインスタンスプロファイルを作成します。インスタンスプロファイルにはロールが含まれ、EC2 インスタンスで実行されるプログラムは一時的な認証情報を取得できます。詳細については、IAM ユーザーガイドのAmazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用して許可を付与するを参照してください。

IAM ロールと IAM ユーザーのどちらを使用するかについては、「IAM ユーザーガイド」の「IAM ロールの作成が適している場合 (ユーザーではなく)」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

AWS でアクセスを制御するには、ポリシーを作成して AWS アイデンティティまたはリソースにアタッチします。ポリシーは AWS のオブジェクトであり、アイデンティティやリソースに関連付けてこれらのアクセス許可を定義します。AWS は、プリンシパル (ユーザー、ルートユーザー、またはロールセッション) がリクエストを行うと、これらのポリシーを評価します。ポリシーでの許可により、リクエストが許可されるか拒否されるかが決まります。大半のポリシーは JSON ドキュメントとして AWSに保存されます。JSON ポリシードキュメントの構造と内容の詳細については、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 AWSJSON ポリシーを使用して、だれ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。

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、リソースで必要なアクションを実行するためのアクセス許可をユーザーに付与するため、IAM ポリシーを作成できます。その後、管理者はロールに IAM ポリシーを追加し、ユーザーはロールを引き継ぐことができます。

IAM ポリシーは、オペレーションの実行方法を問わず、アクションの許可を定義します。例えば、`iam:GetRole` アクションを許可するポリシーがあるとします。このポリシーがあるユーザーは、AWS Management Console、AWS CLI、または AWSAPI からロールの情報を取得できます。

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、IAM ユーザー、ユーザーのグループ、ロールなど、アイデンティティにアタッチできる 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、ユーザーとロールが実行できるアクション、リソース、および条件を制御します。アイデンティティベースのポリシーの作成方法については、「IAM ユーザーガイド」の「IAM ポリシーの作成」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、さらに インラインポリシー または マネージドポリシー に分類できます。インラインポリシーは、単一のユーザー、グループ、またはロールに直接埋め込まれます。マネージドポリシーは、AWS アカウント内の複数のユーザー、グループ、およびロールにアタッチできるスタンドアロンポリシーです。マネージドポリシーには、AWS マネージドポリシーとカスタマー管理ポリシーがあります。マネージドポリシーまたは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 ユーザーガイドのマネージドポリシーとインラインポリシーの比較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。リソースベースのポリシーの例には、IAM ロールの信頼ポリシーや Amazon S3 バケットポリシーがあります。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。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。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。プリンシパルには、アカウント、ユーザー、ロール、フェデレーションユーザー、または AWS のサービスを含めることができます。

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は使用できません。

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、AWS WAF、および Amazon VPC は、ACL をサポートするサービスの例です。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 では、その他の一般的ではないポリシータイプもサポートしています。これらのポリシータイプでは、より一般的なポリシータイプで付与される最大のアクセス許可を設定できます。

- アクセス許可の境界 - アクセス許可の境界は、アイデンティティベースのポリシーによって IAM エンティティ (IAM ユーザーまたはロール) に付与できる最大のアクセス許可を設定する高度な機能です。エンティティにアクセス許可の境界を設定できます。結果として得られるアクセス許可は、エンティティのアイデンティティベースのポリシーとそのアクセス許可の境界の共通部分になります。`Principal` フィールドでユーザーまたはロールを指定するリソースベースのポリシーでは、アクセス許可の境界は制限されません。これらのポリシーのいずれかを明示的に拒否した場合、許可は無効になります。アクセス許可の境界の詳細については、IAM ユーザーガイドのIAM エンティティのアクセス許可の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) - SCP は、AWS Organizations で組織や組織単位 (OU) に最大アクセス許可を指定する JSON ポリシーです。AWS Organizations は、お客様のビジネスが所有する複数の AWS アカウントをグループ化し、一元的に管理するサービスです。組織内のすべての機能を有効にすると、サービスコントロールポリシー SCPを一部またはすべてのアカウントに適用できます。SCP は、各 AWS アカウントのルートユーザーを含む、メンバーアカウント内のエンティティに対するアクセス許可を制限します。組織と SCP の詳細については、「AWS Organizations ユーザーガイド」の「SCP の仕組み」を参照してください。
- セッションポリシー - セッションポリシーは、ロールまたはフェデレーティッドユーザーの一時的なセッションをプログラムで作成する際に、パラメータとして渡す高度なポリシーです。結果として得られるセッションのアクセス許可は、ユーザーまたはロールのアイデンティティベースのポリシーとセッションポリシーの共通部分になります。また、リソースベースのポリシーから許可が派生する場合もあります。これらのポリシーのいずれかを明示的に拒否した場合、許可は無効になります。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許可を理解するのがさらに難しくなります。複数のポリシータイプが関連するとき、リクエストを許可するかどうかを AWS が決定する方法の詳細については、IAM ユーザーガイドのポリシーの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 AWS のその他サービスで IAM がどのように機能するかに関する概要については、IAM ユーザーガイドの「IAM と連携する AWS のサービス」を参照してください。

Amazon MQ は、作成、更新、および削除操作に IAM を使用しますが、ブローカーにはネイティブ ActiveMQ 認証を使用します。詳細については、「ActiveMQ ブローカーの LDAP との統合」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 ポリシーの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 AWS JSON 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。ポリシーアクションの名前は通常、関連する AWS API オペレーショ

ンと同じです。一致する API オペレーションのない許可のみのアクションなど、いくつかの例外があります。また、ポリシーに複数のアクションが必要なオペレーションもあります。これらの追加アクションは、依存アクションと呼ばれます。

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、次のようにカンマで区切ります。

```
"Action": [
      "mq:action1",
      "mq:action2"
```

ワイルドカード (*) を使用して複数のアクションを指定することができます。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 AWS JSON 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、オブジェクトあるいはアクションが適用されるオブジェクトを指定します。ステートメントには、`Resource` または `NotResource` 要素を含める必要があります。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。これは、リソースレベルの許可と呼ばれる特定のリソースタイプをサポートするアクションに対して実行できます。

オペレーションのリスト化など、リソースレベルの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 でのプライマリ 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 (ARN) と AWS のサービスの名前空間」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。複数のリソースを単一の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 AWS JSON 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 (または Condition ブロック) を使用すると、ステートメントが有効な条件を指定できます。Condition 要素はオプションです。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することで、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。

1 つのステートメントに複数の Condition 要素を指定する場合、または 1 つの Condition 要素に複数のキーを指定する場合、AWS では AND 論理演算子を使用してそれらを評価します。単一の条件キーに複数の値を指定する場合、AWS では OR 論理演算子を使用して条件を評価します。ステートメントの許可が付与される前にすべての条件が満たされる必要があります。

条件を指定する際にプレースホルダー変数も使用できます。例えば IAM ユーザーに、IAM ユーザー名がタグ付けされている場合のみリソースにアクセスできる許可を付与することができます。詳細については、IAM ユーザーガイドの「IAM ポリシーの要素: 変数およびタグ」を参照してください。

AWS はグローバル条件キーとサービス固有の条件キーをサポートしています。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/${TagKey}	リクエストで渡されたタグに基づいてアクションをフィルタリングします。	文字列
aws:ResourceTag/${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。	文字列
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。	文字列

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 [Condition element] (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。

```
{
    "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。

- リソースのタグ付け
- 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 AWS アカウント内のエンティティで、特定の許可を持っています。

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 ロールを引き受ける、またはクロスアカウントロール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 または 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。

サービスロール

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。このロールにより、サービスがユーザー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 管理者が、このロールの許可を変更することができます。ただし、これを行うことにより、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。AWS Management Console、AWS CLI、または AWS API を使用してタスクを実行することもできません。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許可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、管理者はそれらの許可が必要な IAM ユーザーまたはグループにそのポリシーを添付します。

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

- ポリシーの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

ポリシーの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を実行すると、AWS アカウント に追加料金が発生する可能性があります。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 マネージドポリシーを使用して開始し、最小特権の許可に移行する – ユーザーとワークロードへの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のために許可を付与する AWS マネージドポリシーを使用します。これらは AWS アカウント で使用できます。ユースケースに応じた AWS カスタマーマネージドポリシーを定義することで、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「AWS ジョブ機能の管理ポリシー」を参照してください。
- 最小特権を適用する – IAM ポリシーで許可を設定するときは、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。例えば、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することができます。また、AWS のサービス などの特定の AWS CloudFormation を介して使用する場合、条件を使用して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。詳細については、「IAM ユーザーガイド」の [IAM JSON policy elements: Condition] (IAM JSON ポリシー要素：条件) を参照してください。
- IAM Access Analyzer を使用して IAM ポリシーを検証し、安全で機能的な許可を確保する - IAM Access Analyzer 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 Access Analyzer 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を作成できるようサポートします。詳細については、「IAM ユーザーガイド」の「IAM Access Analyzer ポリシーの検証」を参照してください。
- 多要素認証 (MFA) を要求する – AWS アカウント で IAM ユーザーまたはルートユーザーを要求するシナリオがある場合は、セキュリティを強化するために MFA をオンにします。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 保護 API アクセスの設定」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。これらの許可は、AWS アカウントの Amazon MQ リソースに関する詳細をリストおよび表示することを許可する必要があります。最小限必要なアクセス許可よりも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、そのポリシーをアタッチしたエンティティ (IAM ユーザーまたはロール) に対しては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 Amazon MQ コンソールを引き続き使用できるようにするため、エンティティに以下の AWS マネージドポリシーもアタッチしてください。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添付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を作成する方法を示します。このポリシーには、コンソールで、または AWS CLI か AWS API を使用してプログラム的に、このアクションを完了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 認証に標準の 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、およびユーザーでの作業を AWS ユーザーに認可するには、IAM ポリシー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 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。

```
{
    "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
UpdateBroker	mq:UpdateBroker
UpdateConfiguration	mq:UpdateConfiguration

Amazon MQ REST API	必要な許可
UpdateUser	mq:UpdateUser

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*

API アクション	リソースタイプ (* 必須)
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
UpdateBroker	ブローカー
UpdateConfiguration	設定*
UpdateUser	ブローカー

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、多くの一般的なユースケースに対するアクセス許可を付与するように設計されているため、ユーザー、グループ、ロールへのアクセス許可の割り当てを開始できます。

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。ユースケース別にカスタマーマネージドポリシーを定義して、マネージドポリシー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 管理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) が更新されます。 は、新しい AWS のサービス が起動されたとき、

または既存のサービスで新しい API AWS オペレーションが使用可能になったときに、 AWS 管理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。

AWS マネージドポリシー: AmazonMQServiceRole ポリシー

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

Amazon MQ の AWS マネージドポリシーの更新に関する詳細を、このサービスがこれらの変更の追跡を開始した以降の分について表示します。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 が AWS マネージド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) サービスリンクロールを使用しています。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。サービスリンクロールは Amazon MQ によって事前に定義されており、サービスがユーザーに代わって AWS のその他サービスを呼び出すために必要なすべての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。定義される許可には、信頼ポリ

シーと許可ポリシーが含まれており、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、まずその関連リソースを削除します。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 のサービスを参照の上、 サービスにリンクされたロール 列が 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リンクロールを使用し、ユーザーに代わって AWS のサービスを呼び出すためにこのサービスリンクロールを使用します。

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。

```
{
    "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きに、Amazon MQ がユーザーに代わって AWS のサービスを呼び出すためのサービスリンク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、同じ方法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、`mq.amazonaws.com` サービス名を使用してサービスリンク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する場合、この同じプロセスを使用して、もう一度ロールを作成できます。

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 を使用したロールの説明の編集はできます。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またはサービスが不要になった場合には、そのロールを削除することをお勧めします。そうすることで、使用していないエンティティがアクティブにモニタリングされたり、メンテナンスされたりすることがなくなります。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、数分待ってから再度オペレーションを実行してください。

AWSServiceRoleForAmazonMQ が使用する Amazon MQ リソースを削除する

- AWS Management Console、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。

IAM を使用してサービスリンクロールを手動で削除する

IAM コンソール、AWS CLI、または AWS API を使用して、AWSServiceRoleForAmazonMQ サービスリンク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 を実行することが認可されていません
- AWS アカウント外のユーザーに 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

AWS Management Console から、アクションを実行する権限がないと通知された場合、管理者に問い合わせ、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* リソースへのアクセスが許可されるように、管理者にポリシーの更新を依頼します。

iam:PassRole を実行することが認可されていません

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せずに、既存のロールをサービスに渡すことが許可されています。そのためには、サービスにロールを渡す許可が必要です。

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、メアリー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 管理者に問い合わせてください。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ユーザーが、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することができます。リソースベースのポリシーまたはアクセス制御リスト (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。
- 所有している AWS アカウント全体のリソースへのアクセス権を提供する方法については、IAM ユーザーガイドの「所有している別の AWS アカウント アカウントへのアクセス権を IAM ユーザーに提供」を参照してください。
- サードパーティーの AWS アカウント にリソースへのアクセス権を提供する方法については、「IAM ユーザーガイド」の「第三者が所有する AWS アカウント へのアクセス権を付与する」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でのロールとリソースベースのポリシーの使用の違いの詳細については、「IAM ユーザーガイド」の「IAM ロールとリソースベースのポリシーとの相違点」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、コンプライアンスプログラムAWS のサービス による対象範囲内のコンプライアンスプログラムを参照し、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、「 でのレポートのダウンロード AWS Artifact」の」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。 では、コンプライアンスに役立つ以下のリソース AWS を提供しています。

- セキュリティとコンプライアンスのクイックスタートガイド – これらのデプロイガイドでは、アーキテクチャ上の考慮事項について説明し、セキュリティとコンプライアンスに重点を置いたベースライン環境 AWS を にデプロイする手順について説明します。
- アマゾン ウェブ サービスにおける HIPAA セキュリティとコンプライアンスのアーキテクチャ — このホワイトペーパーでは、企業が AWS を使用して HIPAA 対象アプリケーションを作成する方法について説明します。

> **Note**
>
> すべて AWS のサービス HIPAA の対象となるわけではありません。詳細については、「HIPAA 対応サービスのリファレンス」を参照してください。

- AWS コンプライアンスリソース – このワークブックとガイドのコレクションは、お客様の業界や地域に適用される場合があります。
- AWS カスタマーコンプライアンスガイド – コンプライアンスの観点から責任共有モデルを理解します。このガイドでは、ガイダンスを保護し AWS のサービス 、複数のフレームワーク (米国国立標準技術研究所 (NIST)、Payment Card Industry Security Standards Council (PCI)、国際標準化機構 (ISO) を含む) のセキュリティコントロールにマッピングするためのベストプラクティスをまとめています。
- 「 デベロッパーガイド」の「 ルールによるリソースの評価」 – この AWS Config サービスは、リソース設定が社内プラクティス、業界ガイドライン、および規制にどの程度準拠しているかを評価します。 AWS Config
- AWS Security Hub – これにより AWS のサービス 、 内のセキュリティ状態を包括的に確認できます AWS。Security Hub では、セキュリティコントロールを使用して AWS リソースを評価し、セキュリティ業界標準とベストプラクティスに対するコンプライアンスをチェックします。サポートされているサービスとコントロールのリストについては、「Security Hub のコントロールリファレンス」を参照してください。
- Amazon GuardDuty — これにより AWS アカウント、疑わしいアクティビティや悪意のあるアクティビティがないか環境を監視することで、、、ワークロード、コンテナ、データに対する潜在的な脅威 AWS のサービス を検出します。 GuardDuty は、特定のコンプライアンスフレームワーク

で義務付けられている侵入検知要件を満たすことで、PCI DSS などのさまざまなコンプライアンス要件への対応に役立ちます。

- AWS Audit Manager – これにより AWS のサービス 、 AWS 使用状況を継続的に監査し、リスクの管理方法と規制や業界標準への準拠を簡素化できます。

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· 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

これはマネージドサービスであり、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 AWS がインフラストラクチャ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 AWS 環境を設計するには、「セキュリティの柱 - AWS Well-Architected Framework」の「インフラストラクチャ保護」を参照してください。

AWS 公開版 API コールを使用して、ネットワーク経由でアクセスします。クライアントは以下をサポートする必要があります。

- Transport Layer Security (TLS) TLS 1.2 および TLS 1.3 をお勧めします。
- DHE (Ephemeral Diffie-Hellman) や ECDHE (Elliptic Curve Ephemeral Diffie-Hellman) などの Perfect Forward Secrecy (PFS) を使用した暗号スイートです。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。

また、リクエストは、アクセスキー ID と、IAM プリンシパルに関連付けられているシークレットアクセスキーを使用して署名する必要があります。または、AWS Security Token Service (AWS STS) を使用して、一時的なセキュリティ認証情報を生成し、リクエストに署名することもできます。

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

- 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
- 認可マップを常に設定する
- 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

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、このガイドの パブリックアクセシビリティが無効化されたブローカーウェブコンソールへのアクセス および 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AWS」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 には承認された承認マップがないため、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

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。

- Configure Additional Broker Settings
- 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うえで重要な部分です。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。AWS 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ツールをいくつか提供します。

トピック

- Amazon MQ 向けの CloudWatch メトリクスへのアクセス
- Amazon CloudWatch を使用した Amazon MQ ブローカーのモニタリング
- AWS CloudTrail を使用した Amazon MQ API コールのロギング
- ログを Amazon CloudWatch Logs に発行するための Amazon MQ の設定

Amazon MQ 向けの CloudWatch メトリクスへのアクセス

Amazon MQ と Amazon CloudWatch は、CloudWatch を使用して ActiveMQ ブローカーとブローカーの送信先 (キューとトピック) のメトリクスを表示し、分析できるように統合されています。Amazon MQ メトリクスは、CloudWatch コンソール、AWS CLI、または CloudWatch CLI を使用して表示および分析することができます。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。

Amazon MQ メトリクスの完全なリストについては、「Monitoring Amazon MQ using CloudWatch」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。

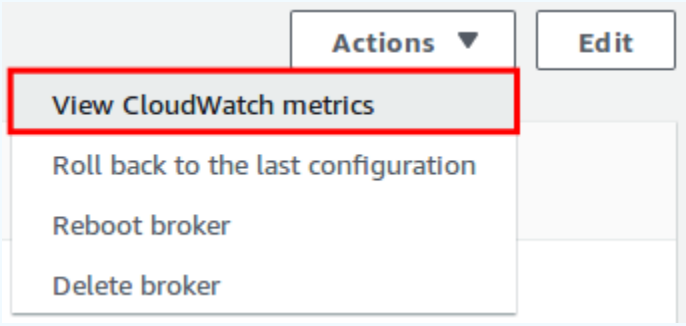
> **Note**
>
> CloudWatch で報告される Amazon MQ メトリクスに料金はかかりません。これらのメトリクスは Amazon MQ サービスの一環として提供されます。
> ActiveMQ ブローカーの場合、CloudWatch は最初の 1000 個の送信先のみをモニタリングします。
> RabbitMQ ブローカーの場合、CloudWatch はコンシューマーの数順に並べられた最初 500 個の送信先のみをモニタリングします。

トピック

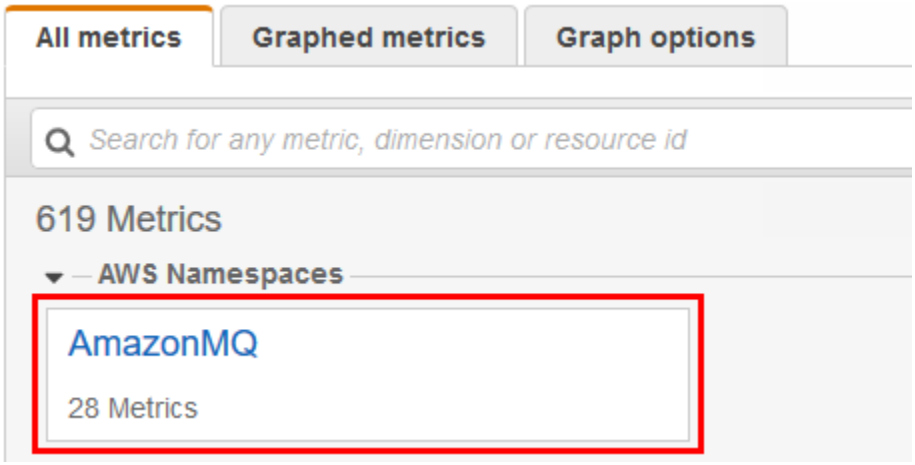
- AWS Management Console
- AWS Command Line Interface
- Amazon CloudWatch API

AWS Management Console

以下の例では、AWS Management Consoleを使用して Amazon MQ 向けの CloudWatch メトリクスにアクセスする方法を説明します。

> **Note**
>
> 既に Amazon MQ コンソールにサインインしている場合は、ブローカーの [Details] (詳細) ページで、[Actions] (アクション)、[View CloudWatch metrics] (CloudWatch メトリクスを表示) の順にクリックします。
>
> 
>

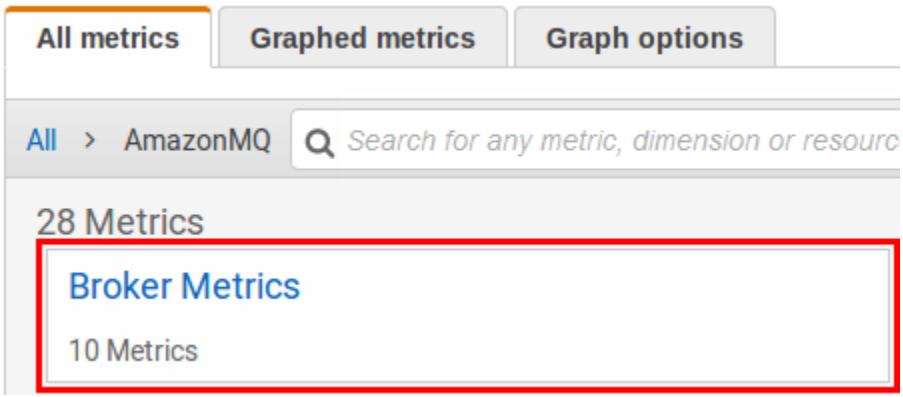
1. CloudWatch コンソールにサインインします。
2. ナビゲーションパネルで [Metrics] を選択します。
3. [AmazonMQ] メトリクスの名前空間を選択します。



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

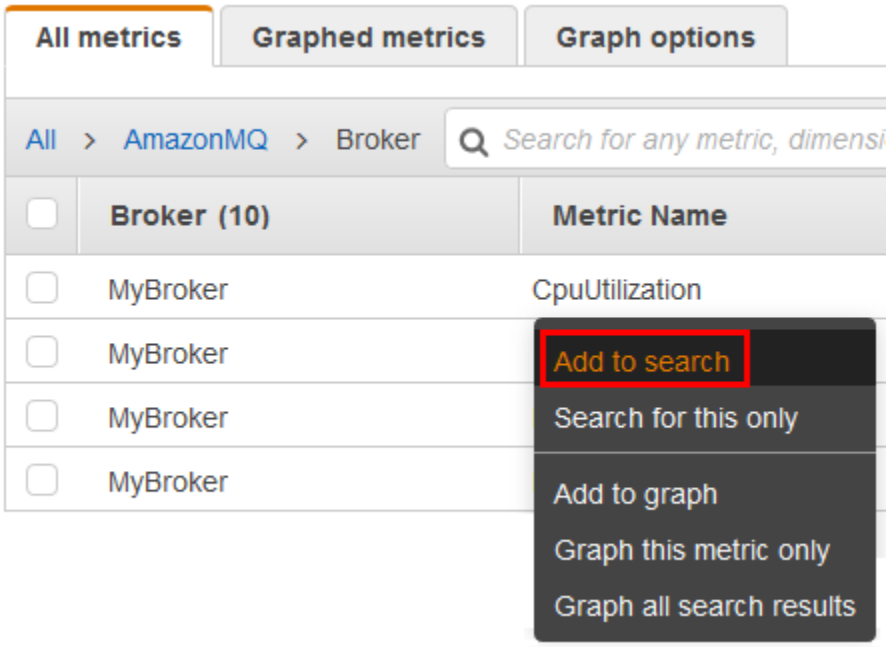
   この例では、[ブローカーのメトリクス] が選択されています。



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。



AWS Command Line Interface

AWS CLI を使用して Amazon MQ メトリクスにアクセスするには、`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。

Amazon CloudWatch API

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。

Amazon CloudWatch を使用した Amazon MQ ブローカーのモニタリング

Amazon MQ と Amazon CloudWatch は、CloudWatch を使用して ActiveMQ ブローカーとブローカーの送信先 (キューとトピック) のメトリクスを表示し、分析できるように統合されていま

す。Amazon MQ メトリクスは、CloudWatch コンソール、AWS CLI、または CloudWatch CLI を使用して表示および分析することができます。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。

詳細については、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。

> **Note**
>
> 次の統計はすべてのメトリクスに対して有効です。
>
> - `Average`
> - `Minimum`
> - `Maximum`
> - `Sum`

`AWS/AmazonMQ` 名前空間には、次のメトリクスが含まれます。

トピック

- Amazon MQ for ActiveMQ ブローカーのロギングとモニタリング
- Amazon MQ for RabbitMQ ブローカーのロギングとモニタリング

Amazon MQ for ActiveMQ ブローカーのロギングとモニタリング

Amazon MQ for ActiveMQ メトリクス

メトリクス	単位	説明
`AmqpMaximumConnections`	Count (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 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

メトリクス	単位	説明
		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	Count (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	Count (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	Count (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	Count (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	Count (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	Count (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	Count (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	Count (カウント)	進行中のトランザクションの総数。
OpenwireMaximumConnections	Count (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	Count (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	Count (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	Count (カウント)	ブローカーに保存されたメッセージの数。
TotalProducerCount	Count (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	Count (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	Count (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	Count (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`	Count (カウント)	送信先にサブスクライブされる消費者の数。
`EnqueueCount`	Count (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	Count (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	Count (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	Count (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	Count (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	Count (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	Count (カウント)	宛先のプロデューサーの数。
QueueSize	Count (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	Count (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	Count (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 ブローカーのロギングとモニタリング

RabbitMQ ブローカーメトリクス

メトリクス	単位	説明
ExchangeCount	Count (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	Count (カウント)	ブローカーで設定されたキューの合計数。
ConnectionCount	Count (カウント)	ブローカーで確立された接続の合計数。
ChannelCount	Count (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	Count (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	Count (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	Count (カウント)	キュー内の準備完了メッセージの合計数。
MessageUnacknowledgedCount	Count (カウント)	キュー内の未承認メッセージの合計数。
PublishRate	Count (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	Count (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	Count (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。Amazon MQ インスタンスタイプの詳細については、「the section called “Amazon MQ for RabbitMQ インスタンスタイプ”」を参照してください。
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。

メトリクス	単位	説明
RabbitMQFdUsed	Count (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。
RabbitMQIOReadAverageTime	Count (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	Count (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。

メトリクス	単位	説明
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。Amazon MQ インスタンスタイプの詳細については、「the section called “Amazon MQ for RabbitMQ インスタンスタイプ”」を参照してください。
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	Count (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。

ディメンション	説明
	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.compute.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	Count (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	Count (カウント)	現在配信可能なメッセージの数。
MessageUnacknowledgedCount	Count (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	Count (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。

AWS CloudTrail を使用した Amazon MQ API コールのロギング

Amazon MQ は、ユーザー、ロール、または AWS のサービスが実行する Amazon MQ コールの記録を提供するサービスである AWS CloudTrail 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

CloudTrail は、AWS アカウントの作成時に有効になります。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、そのアクティビティは、イベント履歴内のその他の AWS サービスイベントと共に 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。証跡を作成することで、AWS アカウントで、実行されているイベントを継続して記録することができます。デフォルトでは、AWS Management Consoleで証跡を作成すると、その証跡はすべてのAWSリージョンに適用されます。追跡は、すべてのAWSリージョンからのイベントをログし、指定された Amazon S3 バケットにログファイルを配信します。その他のAWSのサービスを設定して、CloudTrailログで収集されたデータをより詳細に分析し、それに基づく対応を行うこともできます。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 のサポート対象サービスと統合
- Amazon SNSのCloudTrail通知の設定
- CloudTrailログファイルを複数のリージョンから受け取る
- 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。

- リクエストが、別の AWS のサービスによって行われたか。

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"
}
```

ログを Amazon CloudWatch Logs に発行するための Amazon MQ の設定

Amazon MQ は、さまざまなソースからのログファイルのモニタリング、保存、およびアクセスを行うサービスである Amazon CloudWatch Logs と統合されています。例えば、ブローカーの再起動の通知を受け取るための CloudWatch アラームを設定したり、ActiveMQ ブローカー設定のエラーをトラブルシューティングしたりすることが可能です。CloudWatch Logs 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。

トピック

- Amazon MQ for ActiveMQ ログの設定
- Amazon MQ for RabbitMQ ログの設定

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。

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
- サービス間の混乱した代理の防止
- CloudWatch Logs 設定のトラブルシューティング

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。

```
{
   "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。

以下のリソースベースポリシーは、AWS に `logs:CreateLogStream` および `logs:PutLogEvents` の許可を付与します。

```
{
    "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/*"
        }
    ]
}
```

以下のコマンドにあるように、このリソースベースポリシーは AWS CLI 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しているため、リソースベースのポリシーは、AWS アカウント、リージョン別に一度のみ設定する必要があります。

サービス間の混乱した代理の防止

混乱した代理問題とは、アクションを実行する許可を持たないエンティティが、より高い特権を持つエンティティにそのアクションの実行を強制できるというセキュリティ問題です。AWS では、サービス間でのなりすましが、混乱した代理問題を生じさせることがあります。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、別の顧客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 AWS では、顧客の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を提供しています。これには、アカウント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使用します。

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使用しており、それらが同じポリシーステートメントで使用されるときは、aws:SourceAccount 値と、aws:SourceArn 値のアカウントが同じアカウント ID を使用する必要があります。

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。

```
{
  "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-
east-2:123456789012:broker:MyBroker:b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9"
        }
      }
    }
  ]
}
```

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。

```
{
    "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。

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 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「サービスにリンクされたロールの使用」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。

Amazon MQ のクォータ

このトピックでは、Amazon MQ におけるクォータをリストします。以下のクォータの多くは、特定の AWS アカウントに対して変更することが可能です。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
- Configurations
- Users
- データストレージ
- API スロットリング

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

Configurations

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。

> **Important**
> RabbitMQ ブローカーには適用されません。

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。

> **Important**
> RabbitMQ ブローカーには適用されません。

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。 •

制限	説明
	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。
[Password] (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。 • 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.*.*large` 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

制限	説明
	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.*.*large` 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。

目次

- トラブルシューティング: 一般
  - 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。
  - 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。
  - ブローカーが再起動したのですが、その理由がよくわかりません。
- トラブルシューティング: Amazon MQ for ActiveMQ
  - ログ記録を有効にしても、ブローカーの一般ログまたは監査ログが CloudWatch Logs に表示されません。
  - 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。なぜですか?
  - 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。
  - 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。
- トラブルシューティング: Amazon MQ の RabbitMQ
  - 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。
  - Amazon MQ for 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?
  - ブローカーの Amazon VPC 設定を変更できません。
- トラブルシューティング:Amazon MQ のアクションに必要なコード
  - Amazon MQ for RabbitMQ: 高メモリアラーム
    - 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
    - Amazon MQ メトリクスを使用した高メモリアラームの診断
    - 高メモリアラームへの対応
    - 接続およびチャネルの数の削減
    - クラスターのデプロイで一時停止したキューの同期への対応
    - 単一インスタンスブローカーでの再起動ループへの対応

    - 高メモリアラームの防止
- Amazon MQ for RabbitMQ: 無効な AWS Key Management Service キー
    - INVALID_KMS_KEY の診断と対処
- Amazon MQ for ActiveMQ: 削除された Elastic Network Interface のアラーム
- Amazon MQ for ActiveMQ: ブローカーのメモリ不足アラーム
- Amazon MQ for RabbitMQ: ディスク制限アラーム
    - ディスク制限アラームの診断と対処

トラブルシューティング: 一般

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。
- 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 for ActiveMQ と Amazon MQ for 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 for 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 for 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。

Amazon MQ for ActiveMQ

Amazon MQ for 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>
        ...
```

Amazon MQ for RabbitMQ

Amazon MQ for 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 に発行する前に接続が閉じた場合、このプロセスを繰り返す必要がある場合があります CloudWatch。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 for 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AmazonMQFullAccess、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”」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.t2.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。ブローカーが使用している CPU とメモリの量を確認するには、エンジンタイプに次の CloudWatch メトリクスを使用します。
  - Amazon MQ for 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 for 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。

トラブルシューティング: Amazon MQ for ActiveMQ

このセクションの情報を使用して、Amazon MQ for 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

- ログ記録を有効にしても、ブローカーの一般ログまたは監査ログが CloudWatch Logs に表示されません。
- 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。なぜですか?
- 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。
- 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。

ログ記録を有効にしても、ブローカーの一般ログまたは監査ログが CloudWatch Logs に表示されません。

Logs でブローカーの CloudWatch ログを表示できない場合は、次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 ログを CloudWatch Logs に発行できるようにリソースベースのポリシーを設定しているかどうかを確認します。Amazon MQ が CloudWatch Logs ロググループにログを発行できるようにするには、以下の Logs API アクションへのアクセス権を Amazon MQ CloudWatch に付与するようにリソースベースのポリシーを設定します。
   - `CreateLogStream` – 指定された CloudWatch ロググループのログ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。

ログを Logs に発行するように Amazon MQ for ActiveMQ を設定する方法の詳細については、CloudWatch 「 ログ記録の設定」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。なぜですか?

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、Amazon MQ for ActiveMQ ブローカー CloudWatch の で次のメトリクスをモニタリングします。

- **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。

- Logs で Amazon MQ for ActiveMQ ブローカーの一般的なブローカー CloudWatch 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 for ActiveMQ ブローカーの CloudWatch ログの詳細については、「」を参照してくださいthe section called “CloudWatch Logs での口ギングの構造を理解する”。

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、キューとトピックの承認で `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、すべてのキューとトピックに `activemq-webconsole` グループの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 統合の詳細については、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 for 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

- 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。
- Amazon MQ for 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?
- ブローカーの Amazon VPC 設定を変更できません。

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。

でキューまたは仮想ホストのメトリクスを表示できない場合は CloudWatch、キューまたは仮想ホスト名に空白、タブ、またはその他の非 ASCII 文字が含まれているかどうかを確認します。

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「ディメンション」を参照してください。

Amazon MQ for 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 for 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。

トラブルシューティング:Amazon MQ のアクションに必要なコード

Amazon MQ では、ブローカーが異常な状態で回復が必要な場合、`RebootBroker` などの特定の API オペレーションに対して例外が返されます。例外には、根本原因を特定し、問題に対処してブローカーを回復させるのに役立つ特定のアクション必須コードが含まれます。

次のトピックのリストを使用して、受け取ったアクション必須コードを特定し、問題の解決のために推奨される手順の詳細をご覧ください。

アクション必須コード

- Amazon MQ for RabbitMQ: 高メモリアラーム
- Amazon MQ for RabbitMQ: 無効な AWS Key Management Service キー
- Amazon MQ for ActiveMQ: 削除された Elastic Network Interface のアラーム
- Amazon MQ for ActiveMQ: ブローカーのメモリ不足アラーム
- Amazon MQ for RabbitMQ: ディスク制限アラーム

Amazon MQ for RabbitMQ: 高メモリアラーム

RabbitMQ は、 CloudWatch メトリクス で識別されるブローカーのメモリ使用量が `RabbitMQMemUsed`で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`RabbitMQMemLimit`。 `RabbitMQMemLimit`は Amazon MQ によって設定され、各ホストインスタンスタイプで使用できるメモリを考慮して特別に調整されています。

高メモリアラームが発生した Amazon MQ for 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。メモリ使用率が高いために、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。

メモリ使用率が高いためにスタートアップを完了できない単一インスタンスブローカーは、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。クラスターのデプロイでは、異なるノード上のレプリカ間でのメッセージの同期がキューで一時停止することがあります。キューで同期が一時停止すると、キューからのメッセージの消費が妨げられるため、メモリアラームを解決する際にはこれに個別に対処する必要があります。

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。

このセクションの情報は、ブローカーで発生した RabbitMQ の高メモリアラームの診断と解決に役立ちます。

> **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。

> **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。

トピック

- 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
- Amazon MQ メトリクスを使用した高メモリアラームの診断
- 高メモリアラームへの対応
- 接続およびチャネルの数の削減
- クラスターのデプロイで一時停止したキューの同期への対応
- 単一インスタンスブローカーでの再起動ループへの対応
- 高メモリアラームの防止

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 Management Console 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。ブローカーメトリクスは、CloudWatch コンソールにアクセスするか、 CloudWatch API を使用して表示できます。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ノードメモリに格納されます。チャネルの数が多いと高	

Amazon MQ CloudWatch メトリクス	メモリ使用量が多い理由	
	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。	

高メモリアラームへの対応

特定したコントリビューターごとに、ブローカーの高メモリアラームを軽減して解決するため、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理由	Amazon MQ の推奨	
キュー内のメッセージ数が多すぎます。	次のいずれかを実行します。	

メモリ使用量が多い理由	Amazon MQ の推奨	
	• キューに発行されたメッセージを消費します。 • 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。	
ブローカーで設定されたキューの数が多すぎます。	キューの数を減らします。	
ブローカーで確立された接続の数が多すぎます。	接続の数を減らします。詳細については、「the section called “接続およびチャネルの数の削減”」を参照してください。	
ブローカーで確立されたチャネルの数が多すぎます。	チャネルの数を減らします。詳細については、「the section called “接続およびチャネルの数の削減”」を参照してください。	
ブローカーに接続されたコンシューマーの数が多すぎます。	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。	
メッセージ発行速度が高すぎます。	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。	
クライアント接続試行速度が高すぎます。	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。	

接続およびチャネルの数の削減

Amazon MQ for RabbitMQ ブローカーへの接続は、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 Management Console 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 for 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。

クラスターのデプロイで一時停止したキューの同期への対応

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。

単一インスタンスブローカーでの再起動ループへの対応

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のループ状態にある場合、このセクションで前述した Amazon MQ で推奨されるアクションを適用して、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。

高メモリアラームの防止

特定する要因ごとに、RabbitMQ の高メモリアラームの発生を防止および低減するため、次の一連のアクションを推奨します。

メモリ使用量が多い理由	Amazon MQ の推奨	
キュー内のメッセージ数が多すぎます。	以下の操作を実行します。 • レイジーキューを有効にします。 • 設定を行うか、キューの深度の制限を減らします。	
ブローカーで設定されたキューの数が多すぎます。	設定を行うか、キューの数の制限を減らします。	
ブローカーで確立された接続の数が多すぎます。	設定を行うか、接続の数の制限を減らします。	

メモリ使用量が多い理由	Amazon MQ の推奨	
ブローカーで確立されたチャネルの数が多すぎます。	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。	
ブローカーに接続されたコンシューマーの数が多すぎます。	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。	
クライアント接続試行速度が高すぎます。	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。

ブローカーのインスタンスタイプの一覧については、「the section called “Amazon MQ for RabbitMQ インスタンスタイプ”」を参照してください。

Amazon MQ for RabbitMQ: 無効な AWS Key Management Service キー

Amazon MQ for RabbitMQ は、カスタマー管理 AWS KMS key(CMK) で作成されたブローカーが (KMS) キーが無効であることを検出したときに、INVALID_KMS_KEY AWS Key Management Service 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、 AWS コマンドラインインターフェイス (CLI) 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。

必要な権限がブローカーの KMS キーにまだ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 for 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 for 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 for ActiveMQ: ブローカーのメモリ不足アラーム

Amazon MQ for 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。ブローカーメトリクスは、Amazon CloudWatch コンソールにアクセスするか、 CloudWatch API を使用して表示できます。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。	

Amazon MQ CloudWatch メトリクス	メモリ使用量が多い理由	
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 for ActiveMQ のベストプラクティス」をご覧ください。

Amazon MQ for 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 で識別されるブローカーの空きディスク容量が `RabbitMQDiskFree`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生成します`RabbitMQDiskFreeLimit`。`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できるディスク容量を考慮して定義されています。

ディスク制限が発生した Amazon MQ for RabbitMQ ブローカーは、メッセージが発行されると使用できなくなります。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ときに便利なメトリクスです。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。

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 ユーザーガイドの Amazon MQ	`AWS::AmazonMQ::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::AmazonMQ::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。
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 Premium Support 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、承認の構造を説明した 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。Amazon MQ デベロッパーガイドに対する変更については、「Amazon MQ のドキュメント履歴」を参照してください。

日付	ドキュメントの更新
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。
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のバージョンがサポート終了に達したときを示します。エンジンバージョンのサポートが終了すると、Amazon MQ は、そのバージョンのすべてのブローカーを次にサポートされている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を終了する少なくとも 90 日前に通知します。 バージョンサポートカレンダーとサポート終了を確認するには、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 ブローカーの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中に次のパッチバージョンに更新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 で、マイナーバージョンリリースである RabbitMQ 3.12 がサポートされ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遅延キューとして動作します。

日付	ドキュメントの更新
	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 サーバーリポジトリの RabbitMQ 3.12 リリースノート。RabbitMQ GitHub • RabbitMQ ブローカーの Classic Queue v2 を有効にする • レイジーキューを有効にする 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 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 サーバーリポジトリの RabbitMQ 3.11.28 リリースノート RabbitMQ GitHub • 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 はユーザー名「ゲスト」をサポートしておらず、新しいブローカーを作成するとデフォルトのゲストアカウントが削除されます。Amazon MQ は、お客様が作成した「ゲスト」というアカウントも定期的に削除します。

日付	ドキュメントの更新
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 サーバーリポジトリの RabbitMQ 3.10.25 リリースノート RabbitMQ GitHub • 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 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。

日付	ドキュメントの更新
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 サーバーリポジトリの RabbitMQ 3.11.20 リリースノート RabbitMQ GitHub • 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 サーバーリポジトリの RabbitMQ 3.11.16 リリースノート RabbitMQ GitHub • 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 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

日付	ドキュメントの更新
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 サーバーリポジトリの RabbitMQ 3.9.27 リリースノート RabbitMQ GitHub • 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 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 サーバーリポジトリの RabbitMQ 3.10.20 リリースノート RabbitMQ GitHub • 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 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 3.10.20 で新しいブローカーを作成してください。マイナーバージョンのauto アップグレードオプションを有効にして、最新のバグ修正、セキュリティアップデート、パフォーマンスの向上を自動的に受け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。

日付	ドキュメントの更新
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 サーバーリポジトリの RabbitMQ 3.10.17 リリースノート RabbitMQ GitHub • 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 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。

日付	ドキュメントの更新
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 サーバーリポジトリの RabbitMQ 3.8.34 リリースノート RabbitMQ GitHub • 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 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 サーバーリポジトリの RabbitMQ 3.9.24 リリースノート RabbitMQ GitHub • 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。

日付	ドキュメントの更新
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。
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

日付	ドキュメントの更新
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
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。 • the section called “トラブルシューティング:Amazon MQ のアクションに必要なコード” • 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

日付	ドキュメントの更新
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 サーバーリポジトリの RabbitMQ 3.8.27 リリースノート RabbitMQ GitHub • 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 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。

日付	ドキュメントの更新
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 サーバーリポジトリの RabbitMQ 3.9.13 リリースノート RabbitMQ GitHub • 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 “Amazon MQ for RabbitMQ ブローカーのロギングとモニタリング”

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 サーバーリポジトリの RabbitMQ 3.8.26 リリースノート RabbitMQ GitHub • 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	CloudWatch Logs for Amazon MQ for ActiveMQ ブローカーを設定すると、Amazon MQ は IAM リソースベースのポリシーで 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、混乱した代理問題を防ぐことができます。詳細については、以下を参照してください。 • the section called “サービス間の混乱した代理の防止”

日付	ドキュメントの更新
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 “Amazon MQ for ActiveMQ ブローカーのロギングとモニタリング”
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 サーバーリポジトリの RabbitMQ 3.8.23 リリースノート RabbitMQ GitHub • 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 サーバーリポジトリの RabbitMQ 3.8.22 リリースノート RabbitMQ GitHub • 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、メッセージ単位 (TTL) を使用するキューで特定された問題により、RabbitMQ エンジンバージョン 3.8.17 を一時的に無効にしました。 time-to-live 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 サーバーリポジトリの RabbitMQ 3.8.17 リリースノート RabbitMQ GitHub • RabbitMQ changelog • Amazon MQ for RabbitMQ エンジンバージョンの管理

日付	ドキュメントの更新
2021年7月16日	、、または Amazon MQ API を使用して AWS Management Console AWS CLI、Amazon MQ ブローカーのメンテナンスウィンドウを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
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 リポジトリの RabbitMQ コンシステントハッシュ交換タイプ RabbitMQ GitHub
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。

日付	ドキュメントの更新
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、次を参照してください。 • Amazon MQ for RabbitMQ のブローカーデフォルト
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 が AWS マネージド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 サーバーリポジトリの RabbitMQ 3.8.11 リリースノート RabbitMQ GitHub • 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、「the section called “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 - journalDiskSyncStrategy=periodic の場合にディスク同期を実行する間隔 (ミリ秒)。 • 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 Amazon MQ using CloudWatch」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 • Amazon MQ for ActiveMQ メトリクス 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` と の 2 つの Amazon CloudWatch メトリクスを使用できます`TotalDequeueCount` 。詳細については、「ActiveMQ の送信先 (キューとトピック) メトリクス」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	と の 2 つの新しい Amazon `EstablishedConnectionsCount` CloudWatch メトリクスが利用可能です`InactiveDurableSubscribers` 。詳細については、次を参照してください。 • Monitoring Amazon MQ using CloudWatch • Amazon MQ for ActiveMQ メトリクス
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 • ブローカーの Amazon MQ 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。 • ブローカーの Amazon MQ 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` • `######ConnectionStart###`

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、コスト配分を追跡するのに役立つリソース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、Amazon MQ を SOC 準拠サービス として含めました。
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、Amazon MQ を HIPAA 対応サービス として含めました。

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。
	• ブローカーエンジンのバージョン、インスタンスタイプ、 CloudWatch ログ、メンテナンス設定の編集
	• 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、「Amazon MQ for ActiveMQ メトリクス」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 Logs に発行するように Amazon MQ CloudWatch を設定できます。詳細については、「Configuring Amazon MQ to publish logs to Amazon CloudWatch Logs」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 をログに記録できます。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 ブローカー ConfigurationIdまたは Amazon MQ ブローカーユーザープロパティタイプのプロパティを変更すると、ブローカーはすぐに再起動されます。
	• `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 AWS 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。 • CpuCreditBalance CloudWatch メトリクスはmq.t2.micro ブローカーインスタンスタイプで使用できます。
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*** ページで Connections セクションで、セキュリティグループの名前を選択するか、EC2 コンソール を開きます (VPC コンソールの代わりに)。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「インバウンド接続を有効にする」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。

Amazon MQ のドキュメント履歴

以下の表には、Amazon MQ デベロッパーガイドに対する変更がリストされています。Amazon MQ 機能のリリースと改善については、「Amazon MQ リリースノート」を参照してください。

日付	ドキュメントの更新
2022 年 8 月 22 日	Amazon MQ for ActiveMQ エンジンと Amazon MQ for RabbitMQ エンジンに、別々の親チャプターを作成しました。これらの親チャプターには、エンジンの詳細、チュートリアル、ベストプラクティスが含まれるようになりました。 • Working with Amazon MQ for ActiveMQ • Working with Amazon MQ for RabbitMQ
2022 年 1 月 13 日	ブローカーが異常な状態にあるときに Amazon MQ が返すステータスコードを、ブローカーの診断と復旧に関する詳細情報とともに一覧表示する新しいトラブルシューティングセクションを追加しました。 • the section called “トラブルシューティング:Amazon MQ のアクションに必要なコード”
2021 年 11 月 8 日	Amazon MQ for RabbitMQ ブローカーを使用した Python Pika クライアントの設定について説明する新しいチュートリアルを追加しました。

日付	ドキュメントの更新
	• the section called “Amazon MQ for RabbitMQ でPython Pika を使う”
2021 年 10 月 8 日	Amazon MQ for ActiveMQ と Amazon MQ for RabbitMQ の両方のブローカーエンジンについて、次のトラブルシューティングトピックを追加しました。 • 一部のクライアントは接続できません • the section called “Amazon MQ for 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?” • the section called “ブローカーの Amazon VPC 設定を変更できません。”
2021 年 9 月 22 日	Amazon MQ for ActiveMQ ブローカーに関する一般的な接続、および認証に関する問題のトラブルシューティングについて、次のトピックを追加しました。 • 再起動後にブローカーに接続する • ウェブコンソールでの JSP 例外
2021 年 8 月 12 日	Amazon MQ ブローカーの使用時における一般的な問題のトラブルシューティングについて説明する以下のセクションを追加しました。 • トラブルシューティング
2021 年 7 月 29 日	Amazon MQ for RabbitMQ のバージョン管理と、新しいマイナーおよびメジャーエンジンバージョンのサポート対象化に伴う Amazon MQ ブローカーのアップグレードについて説明する以下のセクションを追加しました。 • the section called “バージョン管理”
2021 年 7 月 21 日	Amazon MQ ブローカーをイベントソース AWS Lambda として に接続する方法を説明する以下のセクションを追加しました。 • Connect your Amazon MQ for ActiveMQ broker to Lambda • Connect your Amazon MQ for RabbitMQ broker to Lambda

日付	ドキュメントの更新
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と、 AWS Management Console、 AWS CLI、または Amazon MQ API を使用してメンテナンスウィンドウを調整する方法について説明します。 • the section called “ブローカーのメンテナンス”
2021 年 6 月 7 日	Amazon MQ for ActiveMQ のバージョン管理と、新しいマイナーおよびメジャーエンジンバージョンのサポート対象化に伴う Amazon MQ ブローカーのアップグレードについて説明する以下のセクションを追加しました。 • the section called “バージョン管理” • the section called “エンジンバージョンのアップグレード”
2021 年 5 月 18 日	Amazon MQ の RabbitMQ ブローカーのデフォルトについて説明する以下のセクションを追加しました。 • the section called “ブローカーのデフォルト”
2021 年 5 月 5 日	Amazon MQ の AWS マネージドポリシーとこれらのポリシーの更新を説明する次のセクションを追加しました。 • the section called “AWS マネージドポリシー”
2021 年 2 月 16 日	Amazon MQ for RabbitMQ に関する以下のチュートリアルセクションを追加しました。 • the section called “一時停止されたキュー同期の解決”

日付	ドキュメントの更新

2020 年 11 月 4 日	• Amazon MQ for RabbitMQ のサポートを文書化するために、以下のセクションを追加しました。
	  • the section called “RabbitMQ ブローカーの作成と接続”
	  • 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”
	  • the section called “RabbitMQ エンジン”
	  • the section called “Amazon MQ for RabbitMQ ログの設定”
	  • the section called “サービスリンクロールの使用”
	• Amazon MQ for RabbitMQ サポートを正確に文書化するために、本ガイドの既存の章とセクションに追加の改訂を行いました。

2019 年 12 月 16 日	• 以下のセクションを追加しました。
	  • 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
	• 以下のセクションの情報を改訂しました。
	  • ブローカー
	  • Broker instance types
	  • Amazon MQ 単一インスタンスブローカー
	  • 高可用性対応の Amazon MQ アクティブ/スタンバイブローカー
	  • Create an ActiveMQ broker
	  • Creating and configuring a broker

2019 年 7 月 19 日	次のセクションでは、暗号化管理に関するコンテンツが変更および追加されました。
	• Amazon MQ のデータ保護
	  • 保管中の暗号化
	  • 転送中の暗号化
	• `EncryptionOptions`

日付	ドキュメントの更新
2019 年 4 月 22 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について、次のセクションが追加されました。 • 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 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。 • 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 • ブローカーの Amazon MQ ネットワーク • Amazon MQ Broker Configuration Parameters
2019 年 1 月 5 日	一部の 1 分あたりのメトリクスに関してドキュメントを改善しました。詳細については、以下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」

2018 年 12 月 19 日	• 以下のセクションを追加しました。
	  • ブローカーの Amazon MQ 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
	  • ######ConnectionStart###
	• networkConnectors 子コレクション要素を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セクションに追加しました。

2018 年 12 月 11 日	ActiveMQ バージョン 5.15.8 の可用性を反映するようにドキュメントを更新しました。
2018 年 12 月 5 日	リソースのタグ付け セクションを追加しました。

日付	ドキュメントの更新
2018 年 10 月 26 日	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 セクションを追加しました。
2018 年 10 月 15 日	「Quotas in Amazon MQ」セクションを更新しました。
2018 年 10 月 1 日	「次のステップ」セクションの情報を訂正しました。

2018 年 9 月 27 日	• ブローカーエンジンのバージョン、インスタンスタイプ、 CloudWatch ログ、メンテナンス設定の編集 セクションを追加しました。
	• 以下のセクションを更新しました。
	  • Create an ActiveMQ broker
	  • Configure Basic Broker Settings

2018 年 9 月 18 日	以下の注記をActiveMQ ブローカーユーザーの作成と管理 セクションに次の注意書きを追加しました。「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。」
2018 年 8 月 31 日	• アクティブ/スタンバイブローカーの用語を明確にしました。詳細については、「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。 • メンテナンスウィンドウの用語を簡素化しました。詳細については、「Amazon MQ ブローカー設定のライフサイクル」を参照してください。 • Configure Additional Broker Settings セクションを書き換えました。 • Amazon MQ for ActiveMQ メトリクス セクションおよび Listing brokers and viewing broker details セクションを更新しました。
2018 年 8 月 15 日	「Create an ActiveMQ broker」セクションの情報を訂正しました。
2018 年 8 月 13 日	パブリックアクセシビリティが無効化されたブローカーウェブコンソールへのアクセス セクションを追加しました。

日付	ドキュメントの更新
2018 年 8 月 2 日	• CloudWatch Logs 設定のトラブルシューティング セクションを追加しました。 • このガイド全体に次の警告が追加されました。 ⚠ 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。」
2018 年 8 月 1 日	以下のセクションの情報を訂正しました。 • CloudWatch Logs でのロギングの構造を理解する • Connect a Java application to your broker
2018 年 7 月 31 日	• 3 分間デモビデオを Getting Started with Amazon MQ セクションに移動しました。 • 3 分間デモビデオを What is Amazon MQ? セクションに追加しました。
2018 年 7 月 30 日	• Configuring Amazon MQ to publish logs to Amazon CloudWatch Logs セクションを追加しました。 • 「Configure Additional Broker Settings」セクションを更新しました。
2018 年 7 月 19 日	• Logging Amazon MQ API calls using CloudTrail セクションを追加しました。
2018 年 7 月 5 日	• 「`authorizationEntry`　」セクションに 認可マップを常に設定する 小要素のクロスリファレンスが追加されました。 • 「ActiveMQ ブローカーの LDAP との統合」の情報を明らかにしました。 • 「API スロットリング」の情報を明らかにしました。

日付	ドキュメントの更新
2018 年 6 月 29 日	• 「Broker instance types」セクションの情報が更新されました。 • 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 セクションを追加しました。
2018 年 6 月 4 日	GitHub、HTML、PDF、および Kindle に加えて、Amazon MQ デベロッパーガイドのリリースノートは RSS フィードとして入手できます。 
2018 年 5 月 29 日	「Working Java Example」セクションに次の変更を行いました。 • STOMP+WSS Java の例を追加しました。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。 • MQTT Java の例が改善されました。 • OpenWire Java の例を改善しました。
2018 年 5 月 24 日	「Working Java Example」セクションの MQTT Java の例のワイヤレベルプロトコルのエンドポイントポートを訂正しました。
2018 年 5 月 22 日	すべての Java 依存関係セクションの情報を訂正しました。
2018 年 5 月 17 日	「Users」セクションの情報を訂正しました。
2018 年 5 月 15 日	「効果的な Amazon MQ パフォーマンスの確保」セクションの情報を訂正しました。
2018 年 5 月 8 日	• 「Amazon MQ REST API 許可リファレンス」を単独のセクションにしました。 • カスタム IAM ポリシーの例が記載された「Amazon MQ ブローカーを作成するために必要な IAM 許可」セクションを作成しました。

日付	ドキュメントの更新
2018 年 5 月 7 日	• このガイド全体で、ブローカーのメンテナンスウィンドウが 2 時間であることを明確にしました。詳細については、「Amazon MQ ブローカー設定のライフサイクル」を参照してください。 • ブローカーの作成に `ec2:CreateNetworkInterface` および `ec2:CreateNetworkInterfacePermission` アクセス権限が必要な理由の説明を追加しました。詳細については、「Amazon MQ の API 認証と認可」を参照してください。
2018 年 5 月 1 日	次のセクションで、アクティブ/スタンバイブローカーのメンテナンスウィンドウに関する情報を明確にしました。 • 高可用性対応の Amazon MQ アクティブ/スタンバイブローカー • Creating and configuring a broker • Creating and applying broker configurations
2018 年 4 月 27 日	以下のセクションを書き直し、Java コード例を、コンシューマーではなくプロデューサーのみで接続プールを使用する推奨事項に一致するように最適化しました。 • 常に接続プールを使用する • 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 • AmazonMQExample.java
2018 年 4 月 26 日	「Working Java Example」セクションに MQTT Java の例を追加しました。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。
2018 年 4 月 4 日	「Amazon MQ との通信」セクションの名前を「Amazon MQ への接続」に変更しました。
2018 年 4 月 3 日	「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」セクションの情報を訂正し明確にしました。

日付	ドキュメントの更新
2018 年 4 月 2 日	「 Amazon MQ でのキューの同時保存とディスパッチ」セクションを「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」セクションに移動しました。

2018 年 3 月 27 日	• 「」セクションの re:Invent 発表ビデオを 3 分間デモビデオWhat is Amazon MQ?に置き換えました。
	• 以下のセクションを再構成しました。
	  • Broker Architecture
	  • Amazon MQ の仕組み
	• 「Amazon MQ ブローカー設定のライフサイクル」を「Broker Architecture」セクションに移動しました。

2018 年 3 月 22 日	本ガイド全体で次の記述を明確化しました。「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。
2018 年 3 月 19 日	本ガイド全体で次の記述を明確化しました。「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。」
2018 年 3 月 15 日	• Amazon MQ Basic elements セクションを再構成しました。
2018 年 3 月 12 日	• 「Amazon MQ のセキュリティベストプラクティス」セクションおよび「Amazon MQ への接続」セクションの情報を明確にし、訂正しました。 • 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 セクションを追加しました。 • 「高度なブローカー設定を構成する」セクションの序文に警告をグループ化しました。

日付	ドキュメントの更新
2018 年 3 月 9 日	• 「認可マップを常に設定する」セクションの情報を訂正し明確にしました。 • 「authorizationEntry」セクションを追加し、「kahaDB」セクションを更新しました。
2018 年 3 月 8 日	• 認可マップを常に設定する セクションを追加しました。 • ブローカーのサフィックスに関する注意事項を「Monitoring Amazon MQ using CloudWatch」セクションに追加しました。
2018 年 3 月 6 日	このガイド全体に次の注意事項を追加しました。 **Note** `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。
2018 年 3 月 1 日	• `CpuCreditBalance` メトリクスを Amazon MQ for ActiveMQ メトリクス セクションに追加しました。 • Amazon MQ 子要素属性 セクションを追加しました。 • 「the section called “許可されている要素”」セクションの要素のリンクを、その属性および子コレクション要素に追加しました。 • の AWS 用語集を修正しました GitHub。
2018 年 2 月 28 日	の画像表示を修正しました GitHub。

日付	ドキュメントの更新
2018 年 2 月 27 日	HTML、PDF、Kindle に加えて、Amazon MQ デベロッパーガイドは で入手できます GitHub。フィードバックを終了するには、右上隅にある GitHub アイコンを選択します。 
2018 年 2 月 26 日	• すべての例と図でリージョンに整合性を持たせました。 • AWS コンソールと製品のウェブページへのリンクを最適化しました。
2018 年 2 月 22 日	以下のセクションの情報を明確にし、訂正しました。 • 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 • 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 • Amazon MQ の API 認証と認可 • ActiveMQ ブローカーの LDAP との統合
2018 年 2 月 21 日	以下のセクションの Java コードを訂正しました。 • Working Java Example • Connect a Java application to your broker • 常に接続プールを使用する
2018 年 2 月 20 日	「Amazon MQ のセキュリティ」セクションおよび「ベストプラクティス」セクションの情報を明確にし、訂正しました。
2018 年 2 月 19 日	• 「常に接続プールを使用する」の Java コードを訂正しました。 • 「ベストプラクティス」セクションと「Amazon MQ のセキュリティ」セクションを再編および拡張しました。

日付	ドキュメントの更新

2018 年 2 月 16 日	• Amazon MQ のセキュリティベストプラクティス セクションを追加しました。
	• 「Amazon MQ への接続」セクションを更新しました。
	• 以下のセクションの Java コードを訂正しました。
	  • Getting Started with Amazon MQ
	  • AmazonMQExample.java

2018 年 2 月 15 日	• 「ベストプラクティス」セクションを再編および拡張しました。
	• 以下のセクションを更新しました。
	  • Amazon MQ の使用を開始するにはどうすればよいですか。
	  • 次のステップ (ご利用開始にあたって)
	  • Related resources

2018 年 2 月 14 日	以下のセクションを更新しました。 • Quotas in Amazon MQ • API スロットリング • Amazon MQ のセキュリティ
2018 年 2 月 13 日	• 「Related resources」セクションを更新しました。 • 「Quotas in Amazon MQ」セクションを更新しました。 • ご意見をお待ちしております セクションを追加しました。
2018 年 1 月 25 日	• 「Java の依存関係を追加する」セクションの「Working Java Example」サブセクションのエラーを修正しました。 • REST オペレーション ID `RebootBroker` の許可が、IAM コンソールで `mq:RebootBroker` として正しく表示されるようになりました。

日付	ドキュメントの更新
2018 年 1 月 24 日	• Amazon MQ Elastic Network Interface を変更または削除しない セクションを追加しました。 • このガイド全体ですべての図が更新されました。 • 本ガイド全体に Amazon MQ REST API リファレンスへのリンクを追加し、「Amazon MQ の API 認証と認可」セクションに特定の REST API へのリンクを追加しました。
2018 年 1 月 19 日	「Amazon MQ for ActiveMQ のリソース」セクションの情報が更新されました。
2018 年 1 月 18 日	「Quotas in Amazon MQ」セクションの情報を訂正し明確にしました。
2018 年 1 月 17 日	永続サブスクリプションより仮想送信先を優先する推奨を復帰させ、説明を改善しました。
2018 年 1 月 11 日	• Amazon MQ デベロッパーガイドは、HTML と PDF に加えて、Kindle 形式でもご利用いただけます。 • 「Amazon MQ の API 認証と認可」および「ステップ 2: ユーザーを作成して AWS 認証情報を取得する」セクションの情報を訂正し明確にしました。
2018 年 1 月 3 日	「`DescribeConfigurationRevision` 」を「Amazon MQ の API 認証と認可」セクションに追加しました。
2017 年 15 月 12 日	「ベストプラクティス」セクションから、永続サブスクリプションに対する推奨事項を削除しました。
2017 年 12 月 8 日	• 「インバウンド接続を有効にする」および「Connecting a Java application to your broker」セクションに「Working Java Example」の前提条件を追加しました。 • このガイド全体に次の注意事項を追加しました。「現在、設定を削除することはできません。」
2017 年 12 月 7 日	• 「AmazonMQExample.java」のコードを強化しました。 • Amazon MQ の API 認証と認可 セクションを追加しました。

日付	ドキュメントの更新

2017 年 5 月 12 日	• 「Monitoring Amazon MQ using CloudWatch」セクションの情報を明確にし、修正しました。
	  • メトリクスの説明を改善しました。
	  • 「Amazon MQ for ActiveMQ メトリクス」および「ActiveMQ ブローカーメトリクスのディメンション」サブセクションを追加しました。
	• 「What is Amazon MQ?」セクションに「Introducing Amazon MQ」動画を追加しました。

2017 年 12 月 4 日	• 「データストレージ」セクションで、次の情報を明確にしました。ブローカーあたりのストレージ容量は 200 GB です。 • 「前提条件」を「Working Java Example」セクションに追加しました (この例を機能させるには、`activemq-client.jar` および `activemq-pool.jar` パッケージが必要です。詳細については、「Connecting a Java application to your broker」を参照してください)。
2017 年 12 月 1 日	• すべてのチュートリアルのスクリーンショットを更新および改善しました。 • 本ガイド全体で次の説明を明確にしました。「設定リビジョン、または ActiveMQ ユーザーを変更しても、変更は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。詳細については、「Amazon MQ ブローカー設定のライフサイクル」を参照してください。

AWS 用語集

AWS の最新の用語については、「AWS の用語集リファレンス」の「AWS 用語集」を参照してください。

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。