aws

开发人员指南

Amazon MQ

Amazon MQ: 开发人员指南

Table of Contents

什么是 Amazon MQ？

Amazon MQ 是一项托管消息代理服务，使其易于迁移到云中的消息代理。消息代理 允许软件应用程序和组件使用各种编程语言、操作系统和正式消息收发协议进行通信。目前，亚马逊 MQ 支持 Apache ActiveMQ Classic 和 RabbitMQ 引擎类型。

Amazon MQ 可与您现有的应用程序和服务配合使用，而无需管理、操作或维护您自己的消息收发系统。

主题

Amazon MQ 与 Amazon SQS 或 Amazon SNS 有何不同？

Amazon MQ 是一项托管式消息代理服务，可兼容许多常见消息代理。我们建议使用 Amazon MQ 从依赖与 JMS 等 API 或 AMQP 0-9-1、AMQP 1.0、MQTT 和 STOMP 等协议兼容的现有消息代理迁移应用程序。 OpenWire

Amazon SQS 和 Amazon SNS 是高度可扩展、易于使用且不需要您设置消息代理的队列和主题服务。我们建议对新的应用程序使用这些服务，这样可以实现几乎不受限制的可扩展性和简单 API。

如何开始使用 Amazon MQ？

- 要使用 Amazon MQ 创建您的第一个代理，请参阅 Getting Started with Amazon MQ。
- 要查找有助于您充分利用 Amazon MQ 的准则和注意事项，请参阅Working with Amazon MQ for ActiveMQ 和Working with Amazon MQ for RabbitMQ
- 要了解有关 Amazon MQ REST API 的信息，请参阅 Amazon MQ REST API 参考。
- 要了解亚马逊 MQ AWS CLI 命令，请参阅《命令参考》中的 Amazon MQ。AWS CLI

我们希望听到您的意见和建议

我们欢迎您提供反馈。要联系我们，请访问 Amazon MQ 论坛。

设置 Amazon MQ

必须先完成以下步骤，然后才能使用 Amazon MQ。

主题

步骤 1：先决条件

注册获取 AWS 账户

如果您没有 AWS 账户，请完成以下步骤来创建一个。

报名参加 AWS 账户

1. 打开 https://portal.aws.amazon.com/billing/signup。
2. 按照屏幕上的说明进行操作。

   在注册时，将接到一通电话，要求使用电话键盘输入一个验证码。

   当您注册时 AWS 账户，就会创建AWS 账户根用户一个。根用户有权访问该账户中的所有 AWS 服务 和资源。作为安全最佳实践，请为用户分配管理访问权限，并且只使用根用户来执行需要根用户访问权限的任务。

AWS 注册过程完成后会向您发送一封确认电子邮件。在任何时候，您都可以通过转至 https://aws.amazon.com/ 并选择我的账户来查看当前的账户活动并管理您的账户。

创建具有管理访问权限的用户

注册后，请保护您的安全 AWS 账户 AWS 账户根用户 AWS IAM Identity Center，启用并创建管理用户，这样您就不会使用 root 用户执行日常任务。

保护你的 AWS 账户根用户

1. 选择 Root 用户并输入您的 AWS 账户 电子邮件地址，以账户所有者的身份登录。AWS Management Console在下一页上，输入您的密码。

   要获取使用根用户登录方面的帮助，请参阅《AWS 登录 用户指南》中的以根用户身份登录。

2. 为您的根用户启用多重身份验证 (MFA)。

   有关说明，请参阅 I A M 用户指南中的为 AWS 账户 根用户启用虚拟 MFA 设备（控制台）。

创建具有管理访问权限的用户

1. 启用 IAM Identity Center

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的启用 AWS IAM Identity Center。

2. 在 IAM Identity Center 中，为用户授予管理访问权限。

   有关使用 IAM Identity Center 目录 作为身份源的教程，请参阅《用户指南》 IAM Identity Center 目录中的使用默认设置配置AWS IAM Identity Center 用户访问权限。

以具有管理访问权限的用户身份登录

- 要使用您的 IAM Identity Center 用户身份登录，请使用您在创建 IAM Identity Center 用户时发送到您的电子邮件地址的登录网址。

  有关使用 IAM Identity Center 用户登录的帮助，请参阅AWS 登录 用户指南中的登录 AWS 访问门户。

将访问权限分配给其他用户

1. 在 IAM Identity Center 中，创建一个权限集，该权限集遵循应用最低权限的最佳做法。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的创建权限集。

2. 将用户分配到一个组，然后为该组分配单点登录访问权限。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的添加组。

第 2 步：创建用户并获取您的 AWS 证书

如果用户想在 AWS 外部进行交互，则需要编程访问权限 AWS Management Console。授予编程访问权限的方式取决于正在访问的用户类型 AWS。

要向用户授予编程式访问权限，请选择以下选项之一。

哪个用户需要编程式访问权限？	目的	方式
人力身份 （在 IAM Identity Center 中管理的用户）	使用临时证书签署向 AWS CLI、 AWS 软件开发工具包或 AWS API 发出的编程请求。	按照您希望使用的界面的说明进行操作。 • 有关的 AWS CLI，请参阅《AWS Command Line Interface 用户指南》 AWS IAM Identity Center中的 “配置 AWS CLI 要使用”。 • 有关 AWS 软件开发工具包、工具和 AWS API，请参阅《软件开发工具包和AWS 工具参考指南》中的 IAM 身份中心身份验证。
IAM	使用临时证书签署向 AWS CLI、 AWS 软件开发工具包或 AWS API 发出的编程请求。	按照 IAM 用户指南中的将临时证书与 AWS 资源配合使用中的说明进行操作。
IAM	（不推荐使用） 使用长期凭证签署向 AWS CLI、 AWS 软件开发工具包或 AWS API 发出的编程请求。	按照您希望使用的界面的说明进行操作。 • 有关信息 AWS CLI，请参阅用户指南中的使用 IAM 用户证书进行身份验证。AWS Command Line Interface • 有关 AWS SDK 和工具，请参阅 S AWS DK 和工具参

哪个用户需要编程式访问权限？	目的	方式
		考指南中的使用长期凭证进行身份验证。 • 有关 AWS API，请参阅 IAM 用户指南中的管理 IAM 用户的访问密钥。

步骤 3：为使用示例代码做好准备

以下教程展示了如何使用与亚马逊 MQ 经纪人合作， AWS Management Console 以及如何以编程方式连接到适用于 ActiveMQ 的亚马逊 MQ 和适用于 RabbitMQ 的亚马逊 MQ 经纪商。要使用 ActiveMQ Java 示例代码，您必须安装 Java 标准版开发工具包并对代码进行一些更改。

您还可以使用 Amazon MQ RES T API AWS 和软件开发工具包，以编程方式创建和管理经纪人。

后续步骤

现在，您已准备好使用 Amazon MQ，可以从创建代理开始。根据您的代理引擎类型，您可以将 Java 应用程序连接到 Amazon MQ for ActiveMQ 代理或使用 RabbitMQ Java 客户端库来将基于 JVM 的应用程序连接到 Amazon MQ for RabbitMQ 代理。

Amazon MQ 入门

这部分将向您展示如何创建 Amazon MQ for ActiveMQ 或 RabbitMQ 代理，以及如何将您的应用程序连接到该代理，从而帮助您进一步熟悉 Amazon MQ。

对于每个代理引擎，创建和连接到代理实例略有不同。选择要使用的以下引擎类型之一，以了解有关创建并连接代理的详细信息。在创建并连接到代理后，您可以找到帮助删除代理的说明。

主题

先决条件

在开始之前，请完成 Setting Up Amazon MQ 中的步骤。

创建并连接到 ActiveMQ 代理

代理 是运行在 Amazon MQ 上的消息代理环境。它是 Amazon MQ 的基本构建块。代理实例类（`m5`、`t3`）和大小（`large`、`micro`）的综合描述是一个代理实例类型（例如 `mq.m5.large`）。有关更多信息，请参阅 代理。

主题

步骤 1：创建 ActiveMQ 代理

第一个也是最常见的 Amazon MQ 任务是创建代理。以下示例说明如何使用创建基本经纪商。 AWS Management Console

1. 登录 Amazon MQ 控制台。
2. 在 Select broker engine (选择代理引擎) 页面上，选择 Apache ActiveMQ。
3. 在 Select deployment and storage (选择部署和存储) 页面的 Deployment mode and storage type (部署模式和存储类型) 部分，执行以下操作：

   a. 选择 Deployment mode (部署模式)（例如 Active/standby broker (主动/备用代理)）。有关更多信息，请参阅 Broker Architecture。

      - 单实例代理由一个可用区中的一个代理组成。代理与您的应用程序以及 Amazon EBS 或 Amazon EFS 存储卷进行通信。有关更多信息，请参阅 Amazon MQ 单实例代理。
      - 高可用性的主动/备用代理由两个不同可用区中的两个代理组成，以冗余对配置。这些代理与您的应用程序以及 Amazon EFS 进行同步通信。有关更多信息，请参阅 用于实现高可用性的 Amazon MQ 主动/备用代理。
      - 有关代理网络示例蓝图的更多信息，请参阅示例蓝图。

   b. 选择 Storage type (存储类型)（例如 EBS）。有关更多信息，请参阅 Storage。

      > **Note**
      >
      > Amazon EBS 在单个可用区内复制数据，但不支持 ActiveMQ 主动/备用部署模式。

   c. 选择下一步。
4. 在 Configure settings (配置设置) 页面的 Details (详细信息) 部分，执行以下操作：

   a. 输入 Broker name (代理名称)。

      > **Important**
      >
      > 请勿在代理名称中添加个人身份信息（PII）或其他机密或敏感信息。代理名称可供其他 AWS 服务（包括日 CloudWatch 志）访问。代理名称不适合用于私有或敏感数据。

   b. 选择 Broker instance type (代理实例类型)（例如 mq.m5.large）。有关更多信息，请参阅 Broker instance types。
5. 在 ActiveMQ Web Console access (ActiveMQ Web 控制台访问) 部分，提供 Username (用户名) 和 Password (密码)。以下限制适用于代理用户名和密码：

   - 用户名只能包含字母数字字符、短划线、句点、下划线和波浪线（- . _ ~）。

- 密码必须至少为 12 个字符，包含至少 4 个唯一字符，并且不得包含逗号、冒号或等号（,:=）。

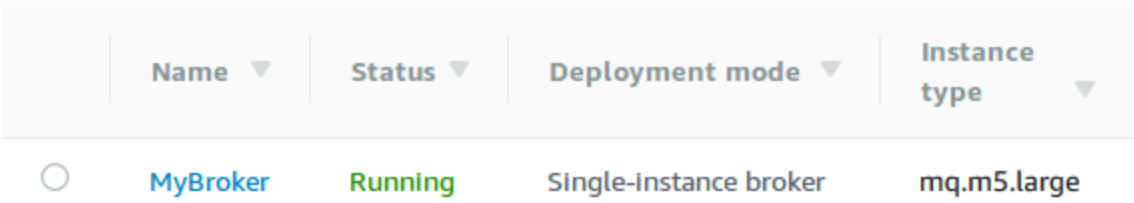
> **Important**
>
> 请勿在代理用户名中添加个人身份信息（PII）或其他机密或敏感信息。其他 AWS 服务（包括 CloudWatch 日志）可以访问经纪人的用户名。代理用户名不适合用于私有或敏感数据。

6. 选择部署。

   当 Amazon MQ 创建您的代理时，会显示 Creation in progress (正在创建) 状态。

   创建代理大约需要 15 分钟。

   成功创建您的代理后，Amazon MQ 会显示 Running (正在运行) 状态。



7. 选择***MyBroker***。

   在该***MyBroker***页面的 Connect 部分，记下您的经纪商的 ActiveMQ 网页控制台网址，例如：

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

   另外，请记下您代理的线级协议终端节点。以下是 OpenWire 终端节点的示例：

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

步骤 2：将 Java 应用程序连接到您的代理

创建 Amazon MQ ActiveMQ 代理后，您可以将应用程序连接到该代理。以下示例演示如何使用 Java Message Service（JMS）创建代理连接、创建队列以及发送消息。有关完整的可用 Java 示例，请参阅Working Java Example。

您可以使用各种 ActiveMQ 客户端连接到 ActiveMQ 代理。我们建议使用 ActiveMQ 客户端。

先决条件

启用 VPC 属性

> **Note**
>
> 您无法禁用现有 Amazon MQ 代理的公共访问权限。

要确保您的代理可以在您的 VPC 中访问，您必须启用 `enableDnsHostnames` 和 `enableDnsSupport` VPC 属性。有关更多信息，请参阅《Amazon VPC 用户指南》中的 VPC 中的 DNS Support。

启用入站连接

1. 登录 Amazon MQ 控制台。
2. 从经纪人列表中，选择您的经纪商的名称（例如 MyBroker）。
3. 在该***MyBroker***页面的 “连接” 部分，记下代理的 Web 控制台 URL 和线级协议的地址和端口。
4. 在 Details (详细信息) 部分的 Security and network (安全与网络) 下，选择您的安全组名称或 。

   此时将显示 EC2 Dashboard 的 Security Groups (安全组) 页面。
5. 从安全组列表中，选择您的安全组。
6. 在页面底部，选择 Inbound (入站)，然后选择 Edit (编辑)。
7. 在 Edit inbound rules (编辑入站规则) 对话框中，为希望公开访问的每个 URL 或终端节点添加规则（以下示例显示如何为代理 Web 控制台执行此操作。

   a. 选择添加规则。

   b. 对于 Type (类型)，选择 Custom TCP (自定义 TCP)。

   c. 对于 Port Range (端口范围)，键入 Web 控制台端口（8162）。

   d. 对于 Source (源)，选择 Custom (自定义)，然后键入您希望能够访问 Web 控制台的系统的 IP 地址（例如 `192.0.2.1`)。

   e. 选择保存。

   您的代理现在可以接受入站连接。

添加 Java 依赖项

将 `activemq-client.jar` 和 `activemq-pool.jar` 程序包添加到 Java 类路径中。以下示例说明了 Maven 项目的 `pom.xml` 文件中的这些依赖关系。

```
<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>
```

有关 `activemq-client.jar` 的更多信息，请参阅 Apache ActiveMQ 文档中的初始配置。

> **Important**
>
> 在以下示例代码中，生产者和使用者在单个线程中运行。对于生产系统（或测试代理实例故障转移），请确保您的创建者和使用者在单独的主机或线程上运行。

创建消息创建者并发送消息

1. 使用代理的终端节点为消息创建者创建 JMS 池连接工厂，然后针对该工厂调用 `createConnection` 方法。

   > **Note**
   >
   > 对于主动/备用代理，Amazon MQ 提供两个 ActiveMQ Web 控制台 URL，但每次只有一个 URL 处于活动状态。同样，Amazon MQ 为每个线级协议提供两个终端节点，但每次每对中只有一个终端节点处于活动状态。`-1` 和 `-2` 后缀表示冗余对。有关更多信息，请参阅 Broker Architecture。
   >
   > 对于线级协议终端节点，您可以允许应用程序使用故障转移传输连接到任一终端节点。

```
// 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**
>
> 消息创建者应始终使用 PooledConnectionFactory 类。有关更多信息，请参阅 始终使用连接池。

2. 创建一个会话，一个名为 MyQueue 的队列和消息创建者。

```
// 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. 创建消息字符串 "Hello from Amazon MQ!"，然后发送消息。

```
// 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. 清理创建者。

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

创建消息使用者并接收消息

1. 使用代理的终端节点为消息创建者创建 JMS 连接工厂，然后针对该工厂调用 `createConnection` 方法。

```
// 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**
>
> 消息使用者绝不 应使用 `PooledConnectionFactory` 类。有关更多信息，请参阅 始终使用连接池。

2. 创建一个会话，一个名为 `MyQueue` 的队列和消息使用者。

```
// 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. 开始等待消息,并在消息到达时收到消息。

```
// 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**
>
> 与 AWS 消息服务（例如 Amazon SQS）不同，消费者经常与经纪人建立联系。

4. 关闭使用者、会话和连接。

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

步骤 3：（可选）连接到 AWS Lambda 函数

AWS Lambda 可以连接并使用来自您的 Amazon MQ 代理的消息。当您将代理连接到 Lambda 时，可以创建事件源映射，从队列中读取消息并同步调用函数。您创建的事件源映射分批从您的代理中读取消息，并以 JSON 对象的形式将它们转换为 Lambda 负载。

将您的代理连接到 Lambda 函数

1. 将以下 IAM 角色权限添加到 Lambda 函数执行角色。

- mq: DescribeBroker
- ec2: CreateNetwork 接口
- ec2: DeleteNetwork 接口
- ec2: DescribeNetwork 接口
- ec2：DescribeSecurity群组
- ec2: DescribeSubnets
- ec2: DescribeVpcs
- 日志：CreateLog组
- 日志：CreateLog直播
- 日志：PutLog事件
- secretsmanager：值 GetSecret

> **Note**
>
> 如果没有必要的 IAM 权限，您的函数将无法从 Amazon MQ 资源中成功读取记录。

2. （可选）如果您创建了一个没有公开可访问性的代理，则必须执行下面其中一项操作以允许 Lambda 连接到您的代理：

   - 为每个公有子网配置一个 NAT 网关。有关更多信息，请参阅《AWS Lambda 开发人员指南》中的VPC 连接函数的互联网和服务访问。
   - 使用 VPC 终端节点在您的 Amazon Virtual Private Cloud（Amazon VPC）和 Lambda 之间创建连接。您的 Amazon VPC 还必须连接到 AWS Security Token Service (AWS STS) 和 Secrets Manager 终端节点。有关更多信息，请参阅《AWS Lambda 开发人员指南》中的为 Lambda 配置接口 VPC 终端节点。

3. 使用 AWS Management Console为 Lambda 函数配置代理作为事件源。您也可以使用该`create-event-source-mapping` AWS Command Line Interface 命令。

4. 为 Lambda 函数编写一些代码来处理从您的代理使用的消息。事件源映射检索的 Lambda 负载取决于代理的引擎类型。以下是适用于 Amazon MQ for ActiveMQ 队列的 Lambda 负载示例。

> **Note**
>
> 在该示例中，`testQueue` 是队列的名称。

```
{
  "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
      }
    ]
  }
}
```

有关将 Amazon MQ 连接到 Lambda、Lambda 为 Amazon MQ 事件源提供支持的选项和事件源映射错误的更多信息，请参阅《AWS Lambda 开发人员指南》中的将 Lambda 与 Amazon MQ 结合使用。

步骤 4：删除代理

如果您不使用亚马逊 MQ 经纪商（也不会预计在不久的将来会使用它），则最佳做法是将其从 Amazon MQ 中删除以降低成本。 AWS

以下示例演示如何使用 AWS Management Console删除代理。

1. 登录 Amazon MQ 控制台。
2. 从经纪商列表中，选择您的经纪商（例如 MyBroker），然后选择删除。
3. 在删除中 ***MyBroker***？ 对话框中，键入，`delete`然后选择 “删除”。

   删除代理大约需要 5 分钟。

后续步骤

现在，您已经创建了一个代理，将一个应用程序连接到了该代理，并发送和接收了一条消息，您可以希望尝试以下操作：

- Creating and configuring a broker（其他设置）
- 编辑代理引擎版本、实例类型、 CloudWatch 日志和维护首选项
- Creating and applying broker configurations
- Listing brokers and viewing broker details
- 创建和管理 ActiveMQ 代理用户
- Rebooting a Broker
- 访问 Amazon MQ 的 CloudWatch 指标

您也可以开始深入了解 Amazon MQ 的最佳实践和 Amazon MQ REST API，然后计划迁移到 Amazon MQ。

创建并连接到 RabbitMQ 代理

代理 是运行在 Amazon MQ 上的消息代理环境。它是 Amazon MQ 的基本构建块。代理实例类（`m5`、`t3`）和大小（`large`、`micro`）的综合描述是一个代理实例类型（例如 `mq.m5.large`）。

主题

步骤 1：创建 RabbitMQ 代理

第一个也是最常见的 Amazon MQ 任务是创建代理。以下示例说明如何使用创建基本经纪商。 AWS Management Console

1. 登录 Amazon MQ 控制台。
2. 在 Select broker engine (选择代理引擎) 页面上，选择 RabbitMQ，然后选择 Next (下一步)。
3. 在 Select deployment mode (选择部署模式) 页面上，选择 Deployment mode (部署模式)，例如 Cluster deployment (集群部署)，然后选择 Next (下一步)。
   - 单实例代理由位于 Network Load Balancer (NLB) 后面的一个可用区中的一个代理组成。代理可与您的应用程序和 Amazon EBS 存储卷进行通信。有关更多信息，请参阅单实例代理。
   - 高可用性的 RabbitMQ 集群部署是由 Network Load Balancer 后面的三个 RabbitMQ 代理节点组成的逻辑分组，每个节点在多个可用区（AZ）之间共享用户、队列和分布式状态。有关更多信息，请参阅用于实现高可用性的集群部署。
4. 在 Configure settings (配置设置) 页面的 Details (详细信息) 部分，执行以下操作：
   a. 输入 Broker name (代理名称)。

      > **Important**
      >
      > 请勿在代理名称中添加个人身份信息（PII）或其他机密或敏感信息。代理名称可供其他 AWS 服务（包括日 CloudWatch 志）访问。代理名称不适合用于私有或敏感数据。

   b. 选择 Broker instance type (代理实例类型)（例如 mq.m5.large）。有关更多信息，请参阅Broker instance types。

> **Note**
>
> 其他设置部分提供了为代理启用 CloudWatch 日志和配置网络访问权限的选项。如果您创建了一个没有公开可访问性的私有 RabbitMQ 代理，则必须选择一个 Virtual Private Cloud（VPC）并配置一个安全组来访问您的代理。

5. 在 Configure settings (配置设置) 页面的 RabbitMQ access (RabbitMQ 访问) 部分，提供 Username (用户名) 和 Password (密码)。以下限制适用于代理程序登录凭证：

   - 用户名只能包含字母数字字符、短划线、句点和下划线（- . _）。此值不得包含任何波浪线（~）字符。Amazon MQ 禁止使用 `guest` 作为用户名。
   - 密码必须至少为 12 个字符，包含至少 4 个唯一字符，并且不得包含逗号、冒号或等号（,:=）。

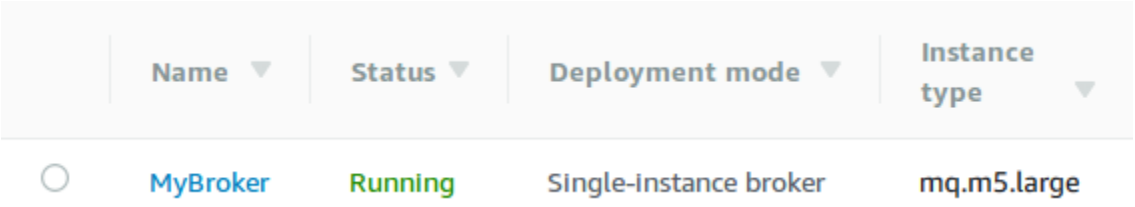
> **Important**
>
> 请勿在代理用户名中添加个人身份信息（PII）或其他机密或敏感信息。其他 AWS 服务（包括 CloudWatch 日志）可以访问经纪人的用户名。代理用户名不适合用于私有或敏感数据。

6. 选择下一步。
7. 在 Review and create (审核和创建) 页面上，您可以查看您的选择并根据需要对其进行编辑。
8. 选择 Create broker (创建代理)。

   当 Amazon MQ 创建您的代理时，会显示 Creation in progress (正在创建) 状态。

   创建代理大约需要 15 分钟。

   成功创建您的代理后，Amazon MQ 会显示 Running (正在运行) 状态。



9. 选择***MyBroker***。

   在***MyBroker***页面上的 Connect 部分，记下您的经纪商的 RabbitMQ 网页控制台网址，例如：

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

另外，请记下您代理的 secure-AMQP 终端节点。以下是一个 amqps 终端节点显示侦听器端口 5671 的示例。

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

步骤 2：将基于 JVM 的应用程序连接到您的代理

创建 RabbitMQ 代理后，您可以将应用程序连接到该代理。以下示例演示如何使用 RabbitMQ Java 客户端库创建与您的代理的连接，创建队列并发送消息。您可以使用各种语言支持的 RabbitMQ 客户端库连接到 RabbitMQ 代理。有关支持的 RabbitMQ 客户端库的更多信息，请参阅 RabbitMQ 客户端库和开发工具。

先决条件

> **Note**
>
> 以下先决条件步骤仅适用于创建的没有公开可访问性的 RabbitMQ 代理。如果您正在创建具有公开可访问性的代理，则可以跳过它们。

启用 VPC 属性

要确保您的代理可以在您的 VPC 中访问，您必须启用 `enableDnsHostnames` 和 `enableDnsSupport` VPC 属性。有关更多信息，请参阅《Amazon VPC 用户指南》中的 VPC 中的 DNS Support。

启用入站连接

1. 登录 Amazon MQ 控制台。
2. 从经纪人列表中，选择您的经纪商的名称（例如 MyBroker）。
3. 在该***MyBroker***页面的 “连接” 部分，记下代理的 Web 控制台 URL 和线级协议的地址和端口。
4. 在 Details (详细信息) 部分的 Security and network (安全与网络) 下，选择您的安全组名称或 。

此时将显示 EC2 Dashboard 的 Security Groups (安全组) 页面。

5. 从安全组列表中，选择您的安全组。
6. 在页面底部，选择 Inbound (入站)，然后选择 Edit (编辑)。
7. 在 Edit inbound rules (编辑入站规则) 对话框中，为希望公开访问的每个 URL 或终端节点添加规则（以下示例显示如何为代理 Web 控制台执行此操作。

   a. 选择添加规则。
   b. 对于 Type (类型)，选择 Custom TCP (自定义 TCP)。
   c. 对于 Source (源)，选择 Custom (自定义)，然后键入您希望能够访问 Web 控制台的系统的 IP 地址（例如 `192.0.2.1`)。
   d. 选择保存。

   您的代理现在可以接受入站连接。

添加 Java 依赖项

如果您使用 Apache Maven 进行自动构建，请将以下依赖项添加到您的 `pom.xml` 文件中。有关 Apache Maven 中的项目对象模型文件的更多信息，请参阅 POM 简介。

```
<dependency>
    <groupId>com.rabbitmq</groupId>
    <artifactId>amqp-client</artifactId>
    <version>5.9.0</version>
</dependency>
```

如果您正在使用 Gradle 进行自动构建，请声明以下依赖项。

```
dependencies {
    compile 'com.rabbitmq:amqp-client:5.9.0'
}
```

导入 **`Connection`** 和 **`Channel`** 类

RabbitMQ Java 客户端使用 `com.rabbitmq.client` 作为其顶级软件包，`Connection` 和 `Channel` API 类分别表示 AMQP 0-9-1 连接和通道。使用前导入 `Connection` 和 `Channel` 类，如以下示例所示。

```
import com.rabbitmq.client.Connection;
```

```
import com.rabbitmq.client.Channel;
```

创建 ConnectionFactory 并连接到您的代理

使用以下示例创建具有给定参数的 ConnectionFactory 类实例。使用 setHost 方法配置您之前记下的代理终端节点。对于 AMQPS 线级连接，请使用端口 5671。

```
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();
```

向交换器发布消息

您可以使用 Channel.basicPublish 将消息发布到交换器。以下示例使用 AMQP Builder 类来构建具有内容类型 plain/text 的消息属性对象。

```
byte[] messageBodyBytes = "Hello, world!".getBytes();
channel.basicPublish(exchangeName, routingKey,
             new AMQP.BasicProperties.Builder()
               .contentType("text/plain")
               .userId("userId")
               .build(),
               messageBodyBytes);
```

> **Note**
>
> 请注意，BasicProperties 是自动生成的持有者类的内部类 AMQP。

订阅队列并接收消息

您可以通过使用 Consumer 接口订阅队列来接收消息。订阅后，消息将在到达时自动传递。

实现 Consumer 的最简单方法是使用子类 DefaultConsumer。DefaultConsumer 对象可以作为 basicConsume 调用的一部分传递以设置订阅，如以下示例所示。

```
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**
>
> 因为我们指定了 autoAck = false，所以必须确认传递到 Consumer 的消息，这在 handleDelivery 方法中完成最为方便，如示例所示。

关闭连接并断开与代理的连接

要断开与您的 RabbitMQ 代理的连接，请关闭通道和连接，如下所示。

```
channel.close();
conn.close();
```

> **Note**
>
> 有关使用 RabbitMQ Java 客户端库的更多信息，请参阅 RabbitMQ Java 客户端 API 指南。

步骤 3：（可选）连接到 AWS Lambda 函数

AWS Lambda 可以连接并使用来自您的 Amazon MQ 代理的消息。当您将代理连接到 Lambda 时，可以创建事件源映射，从队列中读取消息并同步调用函数。您创建的事件源映射分批从您的代理中读取消息，并以 JSON 对象的形式将它们转换为 Lambda 负载。

将您的代理连接到 Lambda 函数

1. 将以下 IAM 角色权限添加到 Lambda 函数执行角色。

   - mq: DescribeBroker
   - ec2: CreateNetworkInterface
   - ec2: DeleteNetworkInterface
   - ec2: DescribeNetworkInterfaces
   - ec2: DescribeSecurityGroups
   - ec2: DescribeSubnets
   - ec2: DescribeVpcs
   - 日志：CreateLogGroup
   - 日志：CreateLogStream
   - 日志：PutLogEvents
   - 秘密管理器：GetSecretValue

   > **Note**
   >
   > 如果没有必要的 IAM 权限，您的函数将无法从 Amazon MQ 资源中成功读取记录。

2. （可选）如果您创建了一个没有公开可访问性的代理，则必须执行下面其中一项操作以允许 Lambda 连接到您的代理：

   - 为每个公有子网配置一个 NAT 网关。有关更多信息，请参阅《AWS Lambda 开发人员指南》中的VPC 连接函数的互联网和服务访问。
   - 使用 VPC 终端节点在您的 Amazon Virtual Private Cloud（Amazon VPC）和 Lambda 之间创建连接。您的 Amazon VPC 还必须连接到 AWS Security Token Service (AWS STS) 和 Secrets Manager 终端节点。有关更多信息，请参阅《AWS Lambda 开发人员指南》中的为 Lambda 配置接口 VPC 终端节点。

3. 使用 AWS Management Console为 Lambda 函数配置代理作为事件源。您也可以使用该create-event-source-mapping AWS Command Line Interface 命令。
4. 为 Lambda 函数编写一些代码来处理从您的代理使用的消息。事件源映射检索的 Lambda 负载取决于代理的引擎类型。以下是 Amazon MQ for RabbitMQ 队列的 Lambda 负载示例。

> **Note**
>
> 在该示例中，`test` 是队列的名称，`/` 是默认虚拟主机的名称。接收消息时，事件源会将消息列在 `test::/` 下。

```
{
  "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=="
      }
    ]
  }
}
```

有关将 Amazon MQ 连接到 Lambda、Lambda 为 Amazon MQ 事件源提供支持的选项和事件源映射错误的更多信息，请参阅《AWS Lambda 开发人员指南》中的将 Lambda 与 Amazon MQ 结合使用。

步骤 4：删除代理

如果您不使用亚马逊 MQ 经纪商（也不会预计在不久的将来会使用它），则最佳做法是将其从 Amazon MQ 中删除以降低成本。 AWS

以下示例演示如何使用 AWS Management Console删除代理。

1. 登录 Amazon MQ 控制台。
2. 从经纪商列表中，选择您的经纪商（例如 MyBroker），然后选择删除。
3. 在删除中 ***MyBroker***？ 对话框中，键入，`delete`然后选择 “删除”。

   删除代理大约需要 5 分钟。

后续步骤

现在，您已经创建了一个代理，将一个应用程序连接到了该代理，并发送和接收了一条消息，您可以希望尝试以下操作：

- 编辑代理引擎版本、实例类型、 CloudWatch 日志和维护首选项
- Listing brokers and viewing broker details
- 创建和管理 ActiveMQ 代理用户
- Rebooting a Broker
- 访问 Amazon MQ 的 CloudWatch 指标

您也可以开始深入了解 Amazon MQ 的最佳实践和 Amazon MQ REST API，然后计划迁移到 Amazon MQ。

管理 Amazon MQ 代理

在以下部分中，您可以找到有关管理和维护 Amazon MQ 代理的说明。

主题

维护 Amazon MQ 代理

Amazon MQ 定期对消息代理的硬件、操作系统或引擎软件进行维护。维护的持续时间有所不同，但最多可持续两小时，具体取决于为消息代理安排的操作。例如，如果您已激活自动次要引擎版本升级或更改了代理实例类型，Amazon MQ 将在下一个计划的维护时段内应用您的更改。

为了最大限度地减少维护时段内的停机时间，建议选择跨多个可用区（AZ）且具有高可用性的代理部署模式。根据您的代理引擎类型，Amazon MQ 提供以下多可用区部署模式。

- Amazon MQ for ActiveMQ – Amazon MQ for ActiveMQ 提供主动/备用部署，以实现高可用性。在主动/备用模式下，Amazon MQ 一次执行一个实例的维护操作，确保至少有一个实例保持可用。此外，您还可以将代理网络维护时段分散在整个周。
- Amazon MQ for RabbitMQ – Amazon MQ for RabbitMQ 提供集群部署，以实现高可用性。在集群部署中，Amazon MQ 一次执行一个节点的维护操作，并始终保持至少两个正在运行的节点。

有关 Amazon MQ 建议的最佳实践以确保您的代理在维护时段期间和之后高效执行操作的更多信息，请参阅以下有关您的代理引擎类型的文档。

- the section called “Amazon MQ for ActiveMQ 最佳实践”
- the section called “Amazon MQ for RabbitMQ 最佳实践”

您可以安排每周进行一次维护，指定最多持续两小时。这将通过 Amazon MQ 设置维护操作时段，以便安排和启动。

您可以在首次创建代理时或通过更新代理首选项来安排维护时段。以下主题介绍如何使用 AWS Management Console AWS CLI、和 Amazon MQ API 调整代理维护窗口。

主题

- 调整代理维护时段

调整代理维护时段

在您选择的维护时段内，Amazon MQ 将执行任何待处理的更改，例如自动次要版本升级。要调整经纪商维护时段，您可以使用 AWS Management Console AWS CLI、或 Amazon MQ API。

> **Important**
>
> 在下一个计划的维护时段开始之前，您最多只能调整代理的维护时段四次。Amazon MQ 会限制四次维护时段调整，以确保关键软件和安全补丁以及重要的硬件升级不会被无限期延迟和延期。
>
> 代理维护时段完成后，Amazon MQ 会重置限制，允许您在下一个维护时段出现之前调整计划。
>
> 调整经纪商维护时段时，代理可用性不会受到影响。

AWS Management Console

要调整经纪商维护时段，请使用 AWS Management Console

1. 登录 Amazon MQ 控制台。
2. 在左侧导航窗格中，选择 Brokers (代理)，然后从列表中选择您要升级的代理。
3. 在代理详细信息页上，选择 Edit (编辑)。
4. 在 Maintenance (维护) 下，执行以下操作。

    a. 对于 Start day（开始日），从下拉列表中选择星期几，例如 Sunday（星期日）。

b. 对于 Start time（开始时间），选择您要为下一个代理维护时段安排的一天中的小时和分钟，例如 12:00。

> **Note**
>
> Start time (开始时间) 选项采用 UTC+0 时区进行配置。

5. 滚动到页面底部并选择 Save (保存)。立即调整维护时段。
6. 在代理详细信息页面上的 Maintenance widnow（维护时段）下，验证是否显示了新的首选计划。

AWS CLI

要调整经纪商维护时段，请使用 AWS CLI

1. 使用 update-broker CLI 命令并指定以下参数，如示例所示。

   - `--broker-id` – Amazon MQ 为代理生成的唯一 ID。您可以通过代理 ARN 解析 ID。例如，给定以下 ARN `arn:aws:mq:us-east-2:123456789012:broker:MyBroker:b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9`，代理 ID 将为 `b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9`。
   - `--maintenance-window-start-time` – 确定以下结构中提供的每周维护时段开始时间的参数。
     - `DayOfWeek` – 星期几，使用以下语法：`MONDAY| TUESDAY | WEDNESDAY | THURSDAY | FRIDAY | SATURDAY | SUNDAY`
     - `TimeOfDay` – 时间，采用 24 小时制。
     - `TimeZone` – （可选）时区，可以采用国家/地区/城市或 UTC 偏移量格式。默认设置为 UTC。

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

2. （可选）使用 describe-broker CLI 命令来验证维护时段是否已成功更新。

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

Amazon MQ API

使用 Amazon MQ API 调整代理维护时段

1. 使用 UpdateBroker API 操作。指定 broker-id 作为路径参数。以下示例假定代理在 us-west-2 区域中。有关可用的 Amazon MQ 端点的更多信息，请参阅《AWS 一般参考》中的 Amazon MQ 端点和限额。

```
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
```

   在请求负载中使用使用 maintenanceWindowStartTime 参数和 WeeklyStartTime资源类型。

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

2. （可选）使用 DescribeBrokerAPI 操作验证维护时段是否已成功更新。 broker-id被指定为路径参数。

```
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
```

升级 Amazon MQ 代理引擎版本

Amazon MQ 定期为所有支持的代理引擎类型提供新的代理引擎版本。新的引擎版本包括安全补丁、错误修复和其他代理引擎改进。

Amazon MQ 根据语义版本控制规范组织版本号，如. X.Y.Z 在 Amazon MQ 实现中，X表示主版本，Y代表次要版本，Z表示补丁版本号。升级有两种类型：

- 主要版本升级 – 当主要引擎版本号更改时发生。例如，从版本 1. 0 升级到版本 2. 0 被视为主要版本升级。
- 次要版本升级-仅次要版本号或补丁引擎版本号发生更改时发生。例如，从版本 1 升级。 5 到版本 1。 6 被视为次要版本升级。

有关各个特定代理引擎类型的主要版本管理和次要版本管理的更多信息，请参阅以下主题。

- the section called “版本管理”
- the section called “版本管理”

您可以随时手动将您的代理升级到下一个支持的主要版本、次要版本或补丁版本。当您开启自动次要版本升级时，Amazon MQ 将在维护时段内将您的代理升级到支持的最新补丁版本。如果您不开启自动次要版本升级，Amazon MQ 会在当前次要版本的支持终止时将您的代理升级到下一个次要版本。

手动和自动版本升级会在计划的维护时段期间或在您重新启动代理之后发生。

以下主题介绍如何手动升级代理引擎版本，以及如何激活自动次要版本升级。

主题

- 手动升级引擎版本
- 自动升级次要引擎版本
- 引擎版本终止支持日历

手动升级引擎版本

要手动将代理的引擎版本升级到新的主要版本或次要版本，您可以使用 AWS Management Console AWS CLI、或 Amazon MQ API。

AWS Management Console

要升级代理的引擎版本，请使用 AWS Management Console

1. 登录 Amazon MQ 控制台。
2. 在左侧导航窗格中，选择 Brokers (代理)，然后从列表中选择您要升级的代理。

3. 在代理详细信息页上，选择 Edit (编辑)。
4. 在 Specifications (规格) 下，对于 Broker engine version (代理引擎版本)，从下拉列表中选择新版本号。
5. 滚动到页面底部并选择 Schedule modifications (计划修改)。
6. 在 Schedule broker modifications (计划代理修改) 页面上，对于 When to apply modifications (何时应用修改) 下，选择以下选项之一。
   - 如果您希望 Amazon MQ 在下一个计划维护时段完成版本升级，请选择 After the next reboot (下次重新启动后)。
   - 如果您想立即重新启动代理并升级引擎版本，请选择 Immediately (立即)。

   > **Important**
   >
   > 您的代理将在重启时脱机。

7. 选择 Apply (应用) 以完成应用更改。

AWS CLI

要升级代理的引擎版本，请使用 AWS CLI

1. 使用 update-broker CLI 命令并指定以下参数，如示例所示。
   - `--broker-id` – Amazon MQ 为代理生成的唯一 ID。您可以通过代理 ARN 解析 ID。例如，给定以下 ARN `arn:aws:mq:us-east-2:123456789012:broker:MyBroker:b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9`，代理 ID 将为 `b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9`。
   - `--engine-version` – 代理引擎要升级到的版本号。

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

2. （可选）如果您想立即升级引擎版本，请使用 reboot-broker CLI 命令重新启动您的代理。

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

   如果您不想重新启动代理和立即应用更改，Amazon MQ 将在下一个计划维护时段内升级代理。

> **Important**
>
> 您的代理将在重启时脱机。

Amazon MQ API

使用 Amazon MQ API 升级代理的引擎版本

1. 使用 UpdateBroker API 操作。指定 `broker-id` 作为路径参数。以下示例假定代理在 `us-west-2` 区域中。有关可用的 Amazon MQ 端点的更多信息，请参阅《AWS 一般参考》中的 Amazon MQ 端点和限额。

   ```
   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
   ```

   在请求负载中使用 `engineVersion` 指定要升级到的代理的版本号。

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

2. （可选）如果您想立即升级引擎版本，请使用 RebootBrokerAPI 操作重启您的代理。 `broker-id`被指定为路径参数。

   ```
   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
   ```

   如果您不想重新启动代理和立即应用更改，Amazon MQ 将在下一个计划维护时段内升级代理。

> **Important**
>
> 您的代理将在重启时脱机。

自动升级次要引擎版本

您可以控制是否在首次创建代理时激活自动次要版本升级，还是通过修改代理首选项来控制。要激活现有代理的自动次要版本升级，您可以使用 AWS Management Console AWS CLI、或 Amazon MQ API。

AWS Management Console

要激活自动次要版本升级，请使用 AWS Management Console

1. 登录 Amazon MQ 控制台。
2. 在左侧导航窗格中，选择 Brokers (代理)，然后从列表中选择您要升级的代理。
3. 在代理详细信息页上，选择 Edit (编辑)。
4. 在 Maintenance (维护) 中，选择 Enable automatic minor version upgrades (启用自动次要版本升级)。

   > **Note**
   >
   > 如果选择了该选项，则无需进行任何更改。

5. 在页面底部选择 Save (保存)。

AWS CLI

要通过激活自动次要版本升级 AWS CLI，请使用 update-broker CLI 命令并指定以下参数。

- `--broker-id` – Amazon MQ 为代理生成的唯一 ID。您可以通过代理 ARN 解析 ID。例如，给定以下 ARN `arn:aws:mq:us-east-2:123456789012:broker:MyBroker:b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9`，代理 ID 将为 `b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9`。
- `--auto-minor-version-upgrade` – 激活自动次要版本升级选项。

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

如果您想为您的代理停用自动次要版本升级，请使用 `--no-auto-minor-version-upgrade` 参数。

Amazon MQ API

要通过 Amazon MQ API 激活自动次要版本升级，请使用 AP UpdateBrokerI 操作。指定 `broker-id` 作为路径参数。以下示例假定代理在 `us-west-2` 区域中。有关可用的 Amazon MQ 端点的更多信息，请参阅《AWS 一般参考》中的 Amazon MQ 端点和限额。

```
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
```

使用请求负载中的 `autoMinorVersionUpgrade` 属性来激活自动次要版本升级。

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

如果您想为您的代理停用自动次要版本升级，请在请求有效负载中设置`"autoMinorVersionUpgrade": "false"`。

引擎版本终止支持日历

Amazon MQ 版本终止支持日历会通知您代理引擎版本何时终止支持。当引擎版本终止支持时，Amazon MQ 会自动将该版本上的所有代理更新到下一个可用版本。在引擎版本终止支持之前，Amazon MQ 会至少在 90 天内发出通知。

要查看版本支持日历，请参阅 ActiveMQ ??? 的亚马逊 MQ 和???适用于 RabbitMQ 的亚马逊 MQ。

代理状态

代理的当前状况由状态 指示。下表列出了 Amazon MQ 代理的状态。

控制台	API	描述
创建失败	CREATION_FAILED	无法创建代理。
正在创建	CREATION_IN_PROGRESS	目前正在创建代理。
正在删除	DELETION_IN_PROGRESS	目前正在删除代理。
正在重启	REBOOT_IN_PROGRESS	目前正在重启代理。
正在运行	RUNNING	代理可以运行。
需采取关键操作	CRITICAL_ACTION_REQUIRED	代理正在运行，但处于降级状态，需要立即执行操作。通过从the section called “故障排除：Amazon MQ 操作所需代码”中的列表内选择操作所需代码，您可以找到解决问题的说明。

列出 Amazon MQ 代理并查看代理详细信息

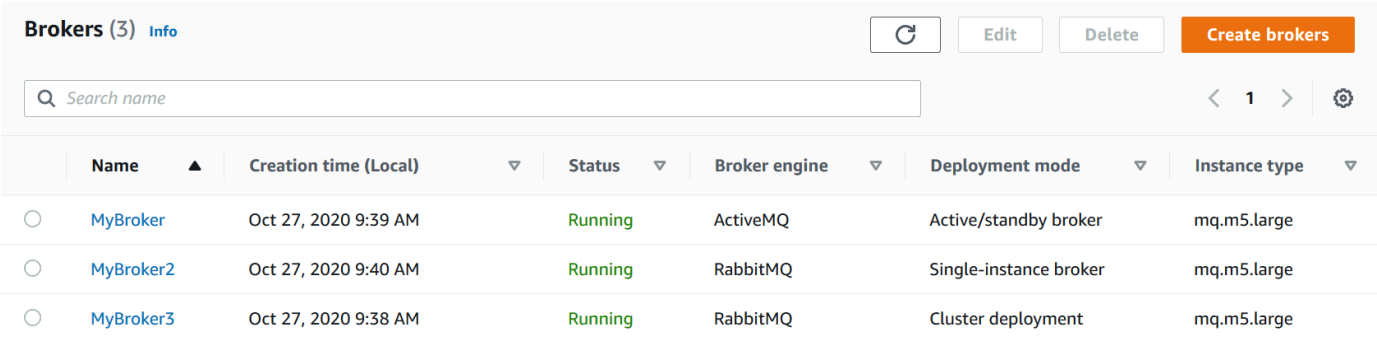
在您请求 Amazon MQ 创建代理时，创建过程可能需要大约 15 分钟。

以下示例演示了如何使用 AWS Management Console通过列出当前区域中您的代理来确认您的代理的存在。

列出代理并查看代理详细信息

1. 登录 Amazon MQ 控制台。

   列出您当前区域中的代理。

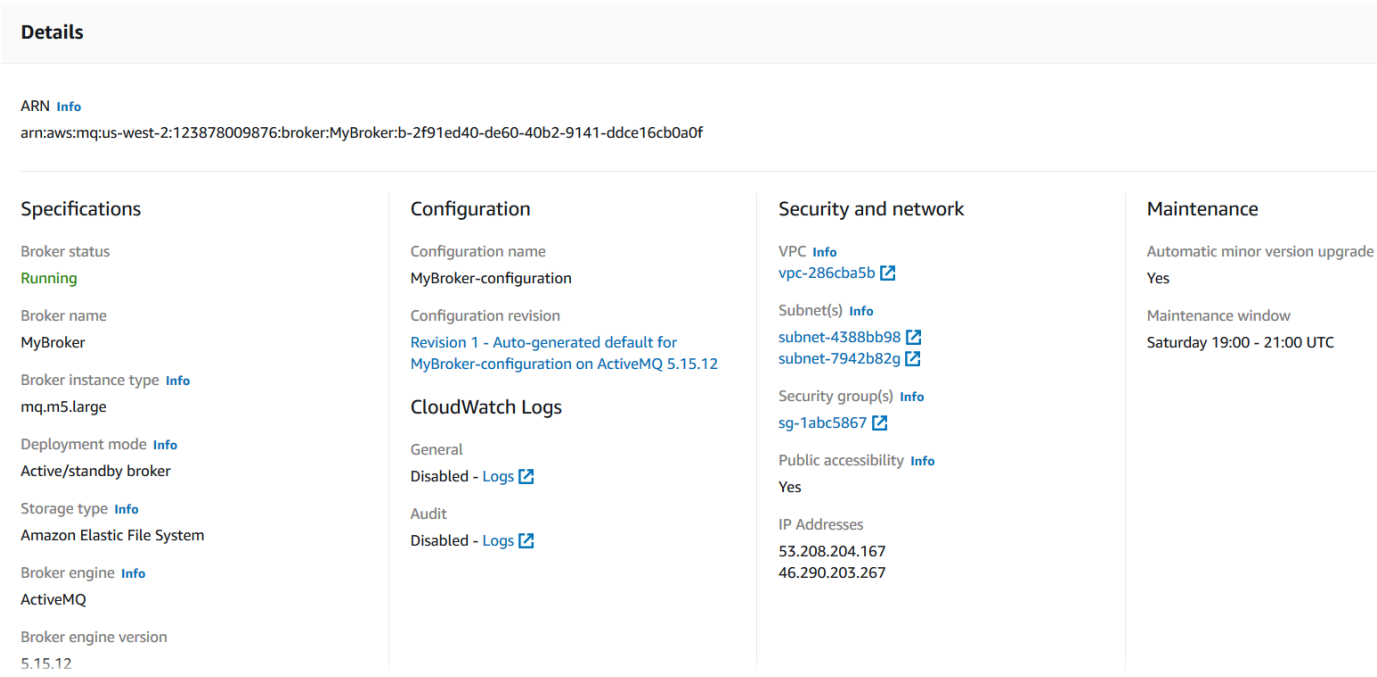


系统会显示每个代理的以下信息：

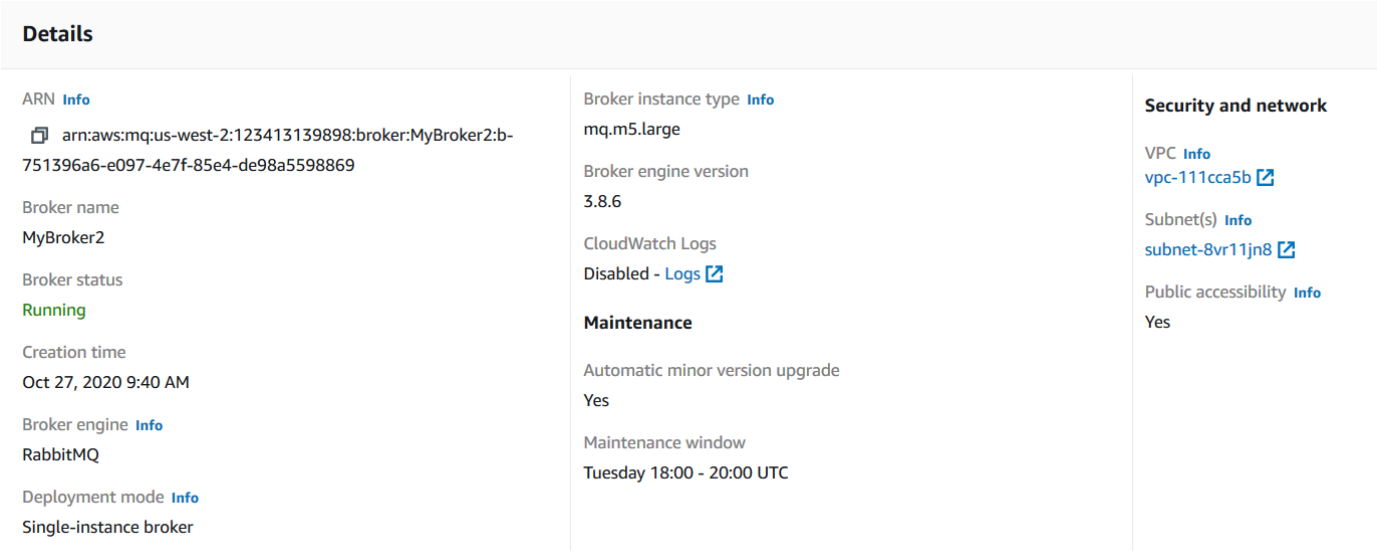
- 名称
- Creation date (创建日期)
- 状态
- Deployment mode (部署模式)
- 实例类型

2. 选择您的代理的名称。

对于 ActiveMQ 代理，***MyBroker*** 页面上会为您的代理显示已配置的 Details (详细信息)：

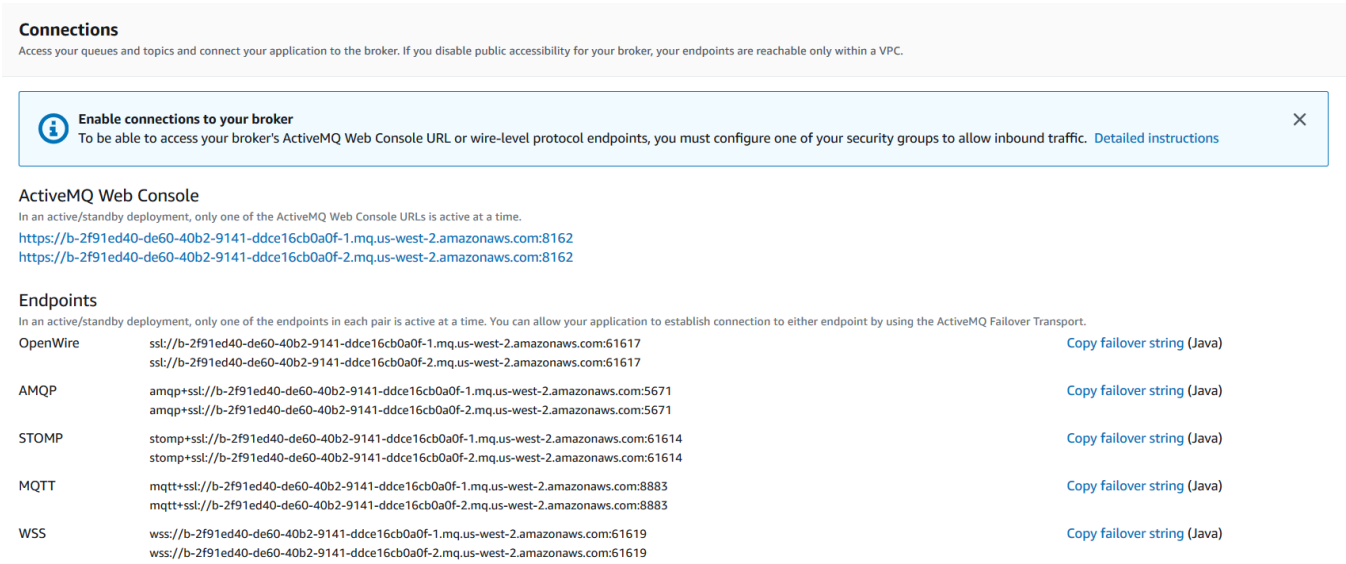


对于 Amazon MQ for RabbitMQ 代理，您可以在 ***MyBroker2*** 页面上的 Details (详细信息) 部分中查看选定的设置，如下所示。

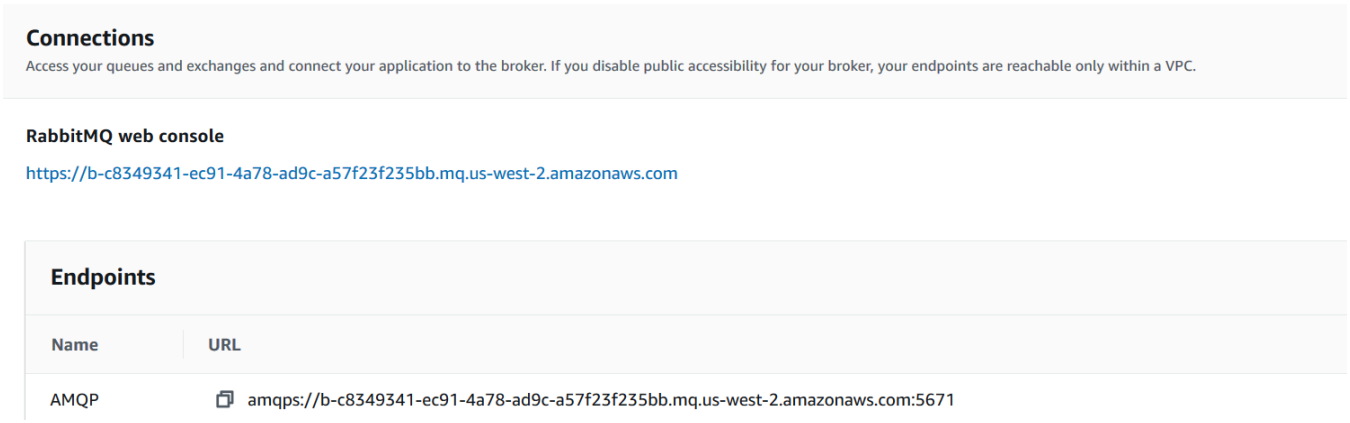


在 Details (详细信息) 部分下，将显示以下信息：

- 在 Connections (连接) 部分中，对于 Amazon MQ for ActiveMQ 代理，将显示 Web 控制台 URL 和线级协议终端节点。



在 Connections (连接) 部分中，对于 Amazon MQ for RabbitMQ 代理，将显示 Web 控制台 URL 和安全 AMQP 终端节点。



- 对于 Amazon MQ for ActiveMQ 代理，在 Users (用户) 部分中，Users (用户) 已与代理关联

> **Important**
>
> Amazon MQ for RabbitMQ 代理不支持通过 AWS Management Console 和 Amazon MQ API 管理用户。

访问不可公开访问的代理 Web 控制台

如果您禁用了代理的公开可访问性，则必须执行下列步骤才能访问代理的 Web 控制台。

> **Note**
>
> VPC 和安全组的名称特定于以下示例。

先决条件

要执行下列步骤，您必须先配置以下内容：

- VPC
  - Amazon MQ 代理所附加到的 VPC，没有互联网网关，名为 `private-vpc`。
  - 第二个 VPC，具有 Internet 网关，名为 `public-vpc`。
  - 两个 VPC 都必须进行连接（例如通过 VPC 对等连接），这样公有 VPC 中的 Amazon EC2 实例就可以与私有 VPC 中的 EC2 实例通信。
  - 如果使用的是 VPC 对等连接，这两个 VPC 的路由表必须配置为对等连接。

- 安全组
  - 用于创建 Amazon MQ 代理的安全组，名为 `private-sg`。
  - 第二个用于 `public-vpc` VPC 中的 EC2 实例的安全组，名为 `public-sg`。
  - `private-sg` 必须允许来自 `public-sg` 的入站连接。我们建议将此安全组限制为端口 8162（对于 ActiveMQ）以及端口 443（对于 RabbitMQ）。
  - `public-sg` 必须允许端口 22 上来自您的计算机的入站连接。

访问不可公开访问的代理的 Web 控制台

1. 在 `public-vpc` 中创建 Linux EC2 实例（如有必要，请包含公有 IP）。
2. 要验证 VPC 的配置是否正确，请建立到 EC2 实例的 `ssh` 连接，并将 `curl` 命令与您代理的 URI 结合使用。
3. 从您的计算机中，使用私有密钥文件的路径和公有 EC2 实例的 IP 地址创建到 EC2 实例的 `ssh` 隧道。例如：

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

   在您的计算机上启动转发代理服务器。
4. 在您的计算机上安装代理客户端，如 FoxyProxy。
5. 使用以下设置配置您的代理客户端：

   - 对于代理类型，请指定 `SOCKS5`。
   - 对于 IP 地址、DNS 名称和服务器名称，请指定 `localhost`。
   - 对于端口，请指定 `8080`。
   - 删除任何现有的 URL 模式。
   - 对于 URL 模式，请指定 `*.mq.*.amazonaws.com*`
   - 对于连接类型，请指定 `HTTP(S)`。

   在您启用代理客户端后，便可以在您的计算机上访问 Web 控制台了。

重启 Amazon MQ 代理

要对代理应用新配置，您可以重启代理。

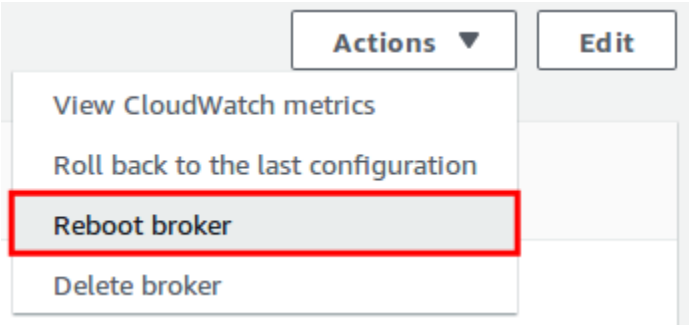
> **Note**
>
> 如果您的 ActiveMQ 代理无法响应，您可以重启代理以从故障状态恢复。

以下示例演示如何使用AWS Management Console重启 Amazon MQ 代理。

要重启 Amazon MQ 代理，请执行以下操作：

1. 登录 Amazon MQ 控制台。
2. 从代理列表中选择您的代理的名称（例如 MyBroker)。
3. 在 ***MyBroker*** 页面上，依次选择 Actions、Reboot broker。

> **Important**
>
> 单实例代理程序在重启时将处于脱机状态。集群代理将可用，但一次只能重启一个节点。



4. 在 Reboot broker 对话框中，选择 Reboot。

   重启一个代理大约需要 5 分钟。如果重启包括实例大小更改或在队列深度较高的代理上执行，则重启过程可能需要更长的时间。

删除 Amazon MQ 代理

如果您不使用 Amazon MQ 代理（并且估计未来近期也不会使用代理），最佳实践是将其从 Amazon MQ 中删除以减少AWS成本。

以下示例演示如何使用AWS Management Console删除代理。

删除 Amazon MQ 代理

1. 登录 Amazon MQ 控制台。
2. 从代理列表中，选择您的代理（例如 MyBroker），然后选择 Delete (删除)。
3. 在 Delete ***MyBroker***? (是否删除 MyBroker？) 对话框中，键入 `delete`，然后选择 Delete (删除)。

   删除代理大约需要 5 分钟。

管理 Amazon MQ 代理配置

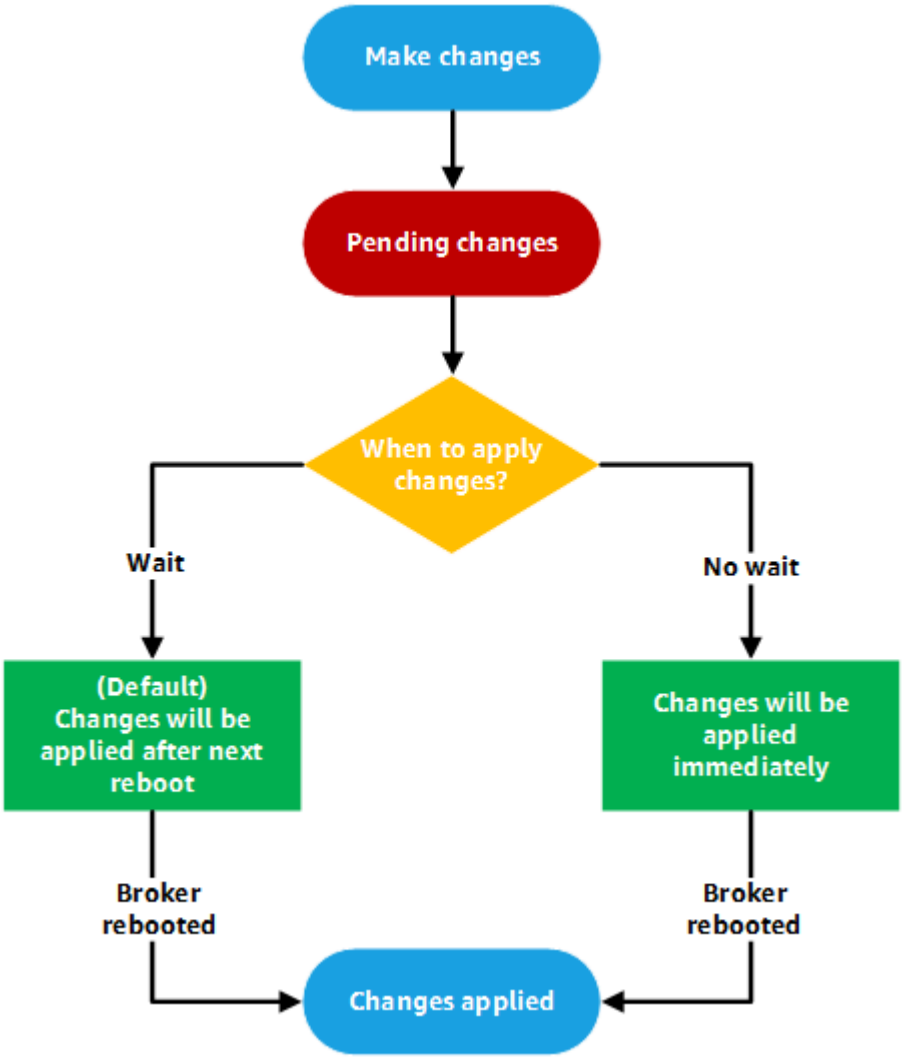
配置包含您的代理的所有设置。您可以先创建配置，然后创建代理。之后，您可以将配置应用于一个或多个代理

Amazon MQ 代理配置生命周期

对配置修订版或 ActiveMQ 用户进行更改不会立即应用更改。要应用更改，必须等待下一维护时段或者重启代理。有关更多信息，请参阅Amazon MQ 代理配置生命周期。

下图说明了配置生命周期。

> **Important**
>
> 下一个计划维护时段会触发重启。如果在下一个计划维护时段之前重新启动代理，则会在重新启动后应用更改。



对于 ActiveMQ，配置包含您的代理的所有设置（采用 XML 格式，类似于 ActiveMQ 的 `activemq.xml` 文件）。有关创建、应用和编辑 ActiveMQ 代理配置的更多信息，请参阅Creating and applying broker configurations。

对于 RabbitMQ，配置包含您的代理的所有设置（采用 Cuttlefish 格式）。有关创建、应用和编辑 RabbitMQ 代理配置的更多信息，请参阅Creating and applying broker configurations

实例类型

代理实例类（`m5`、`t3`）和大小（`large`、`micro`）的综合描述是一个代理实例类型（例如 `mq.m5.large`）。下表列出了每种支持引擎类型的可用 Amazon MQ 代理实例类型。

主题

- Amazon MQ for ActiveMQ 实例类型

- Amazon MQ for RabbitMQ 实例类型

Amazon MQ for ActiveMQ 实例类型

> **Important**
>
> 您只能将 Amazon EBS 与 mq.m5 代理实例类型系列配合使用。有关更多信息，请参阅 Storage。

实例类型	vCPU	内存 (GiB)	网络性能	推荐用途
mq.t2.micro	1	1	低	评估
mq.t3.micro	2	1	低	评估
mq.m4.large	2	8	中	生产
mq.m5.large	2	8	高	生产
mq.m5.xlarge	4	16	高	生产
mq.m5.2xlarge	8	32	高	
mq.m5.4xlarge	16	64	高	

有关吞吐量注意事项的更多信息，请参阅选择正确的代理实例类型以实现最佳吞吐量。

Amazon MQ for RabbitMQ 实例类型

> **Important**
>
> 您不能将代理从 mq.m5. 实例类型降级为 mq.t3.micro 实例类型。

实例类型	vCPU	内存 (GiB)	网络性能	应用场景
mq.t3.micro	2	1	低	评估 **Important** mq.t3.micro 实例类型不支持集群部署。
mq.m5.large	2	8	高	生产
mq.m5.xlarge	4	16	高	生产
mq.m5.2xlarge	8	32	高	
mq.m5.4xlarge	16	64	高	

为 资源添加标签

Amazon MQ 支持资源标记以帮助跟踪您的成本分配。您可以在创建资源时标记资源，也可以通过查看资源的详细信息来标记该资源。

主题

- 成本分配的标记
- 在 Amazon MQ 控制台中管理标签
- 使用 Amazon MQ API 操作进行管理

成本分配的标记

要组织并标识您的 Amazon MQ 资源以进行成本分配，您可以添加用于标识代理或配置目的的元数据标签。这在您拥有许多代理时尤其有用。您可以使用成本分配标签组织 AWS 账单，以反映您自己的成本结构。要执行此操作，请注册以获取 AWS 账户账单来包含标签键和值。有关更多信息，请参阅《AWS Billing用户指南》中的设置月度成本分配报告。

例如，您可以添加表示 Amazon MQ 资源的成本中心和用途的标签：

资源	密钥	值
Broker1	Cost Center	34567
	Stack	Production
Broker2	Cost Center	34567
	Stack	Production
Broker3	Cost Center	12345
	Stack	Development

此标记方案能让您将执行相关任务的两个代理分组到同一成本中心，同时使用不同的成本分配标签标记不相关的代理。

在 Amazon MQ 控制台中管理标签

向新资源添加标签

Amazon MQ 允许您在创建资源时向其添加标签。您可以在 Amazon MQ 控制台中快速向您正在创建的资源添加标签。

要在创建新代理时添加标签，请执行以下操作：

1. 从 Create a broker (创建代理) 页面中，选择 Additional settings (其他设置)。
2. 在 Tags (标签) 下面，选择 Add tag (添加标签)。
3. 输入 Key (键) 和 Value (值) 对。

4. （可选）选择 Add tag (添加标签) 以向代理添加多个标签。
5. 选择 Create broker (创建代理)。

要在创建配置时添加标签，请执行以下操作：

1. 从 Create configuration (创建配置) 页面中，选择 Advanced (高级)。
2. 在 Create configuration (创建配置) 页面上的 Tags (标签) 下，选择 Add tag (添加标签)。
3. 输入 Key (键) 和 Value (值) 对。
4. （可选）选择 Add tag (添加标签) 以向配置添加多个标签。
5. 选择 Create configuration (创建配置)。

查看和管理现有资源的标签

Amazon MQ 可让您在 Amazon MQ 控制台中查看和管理资源的标签。您可以通过在单个资源的详细信息页面上编辑标签来管理该资源的标签。要在 Amazon MQ 资源上编辑标签，请执行以下操作：

1. 在 Amazon MQ 控制台中选择 Brokers (代理) 或 Configurations (配置)。

   在 Tags (标签) 部分中，查看该资源的现有标签。
2. 要添加新标签或管理现有标签，请选择 Edit (编辑)（或者，如果没有现有标签，则选择 Create tag (创建标签)）。
3. 更新资源的标签：

   - 要修改现有标签，请编辑 Key (键) 和 Value (值)。
   - 要删除现有标签，请选择 Remove (删除)。
   - 要添加新标签，请选择 Add tag (添加标签)，然后输入 Key (键) 和 Value (值)。
4. 选择 Save。

使用 Amazon MQ API 操作进行管理

Amazon MQ 能让您使用 REST API 查看和管理资源的标签。

有关更多信息，请参阅 Amazon MQ REST API 参考。

使用 Amazon MQ for ActiveMQ

利用 Amazon MQ，可以轻松使用适合您的需求的计算和存储资源创建消息代理。您可以使用AWS Management Console、Amazon MQ REST API 或 AWS Command Line Interface 创建、管理和删除代理。

这部分将介绍 ActiveMQ 和 RabbitMQ 引擎类型的消息代理的基本要素，列出可用的 Amazon MQ 代理实例类型及其状态，并概述代理架构和配置选项。

要了解有关 Amazon MQ REST API 的信息，请参阅 Amazon MQ REST API 参考。

主题

ActiveMQ 引擎

此部分将介绍 ActiveMQ 代理的基本元素，概述 ActiveMQ 代理架构选项，说明代理配置参数并提供使用 Java Message Service（JMS）的有效示例。

主题

基本元素

本部分介绍对理解 Amazon MQ 上的 ActiveMQ 必不可少的主要概念。

主题

- 代理
- 代理实例类型
- 配置
- 用户
- Storage

代理

代理 是运行在 Amazon MQ 上的消息代理环境。它是 Amazon MQ 的基本构建块。代理实例类（`m5`、`t3`）和大小（`large`、`micro`）的综合描述是一个代理实例类型（例如 `mq.m5.large`）。有关更多信息，请参阅 Broker instance types。

- 单实例代理由一个可用区中的一个代理组成。代理与您的应用程序以及 Amazon EBS 或 Amazon EFS 存储卷进行通信。
- 主动/备用代理由两个不同可用区中的两个代理组成，配置为冗余对。这些代理与您的应用程序以及 Amazon EFS 进行同步通信。

有关更多信息，请参阅 Broker Architecture。

您可以启用自动次要版本升级以在 Apache 发布代理引擎的新次要版本时自动升级到新次要版本。自动升级在维护时段内发生，该维护时段使用星期几、几点（24 小时格式）和时区（默认为 UTC）定义。

有关创建和管理代理的信息，请参阅以下内容：

- Creating and configuring a broker
- 代理
- Broker statuses

支持的线级协议

您可以访问您的代理，方法是使用 ActiveMQ 支持的任何编程语言并通过为以下协议明确启用 TLS：

- AMQP
- MQTT
- MQTT 结束了 WebSocket

- OpenWire
- STOMP
- 大吃一惊 WebSocket

Attributes

ActiveMQ 代理具有几个属性，例如：

- 名称 (`MyBroker`)
- ID (`b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9`)
- Amazon Resource Name（ARN）(`arn:aws:mq:us-east-2:123456789012:broker:MyBroker:b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9`)
- ActiveMQ Web 控制台 URL (`https://b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-east-2.amazonaws.com:8162`)

  有关更多信息，请参阅 Apache ActiveMQ 文档中的 Web 控制台。

  > **Important**
  >
  > 如果您指定的授权映射不包含在 `activemq-webconsole` 组中，您无法使用 ActiveMQ Web 控制台，因为该组未获得授权向 Amazon MQ 代理发送消息或接收来自该代理的消息。

- 线级协议终端节点:
  - `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**
    >
    > 这是一个 OpenWire 端点。

  - `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`

有关更多信息，请参阅 Apache ActiveMQ 文档中的配置传输。

> **Note**
>
> 对于主动/备用代理，Amazon MQ 提供两个 ActiveMQ Web 控制台 URL，但每次只有一个 URL 处于活动状态。同样，Amazon MQ 为每个线级协议提供两个终端节点，但每次每对中只有一个终端节点处于活动状态。`-1` 和 `-2` 后缀表示冗余对。

有关代理属性的完整列表，请参阅《Amazon MQ REST API 参考》中的以下内容：

- REST 操作 ID：Broker
- REST 操作 ID：Brokers
- REST 操作 ID：Broker Reboot

代理实例类型

> **Important**
>
> 您只能将 Amazon EBS 与 `mq.m5` 代理实例类型系列配合使用。有关更多信息，请参阅 Storage。

实例类型	vCPU	内存 (GiB)	网络性能	注意
`mq.t2.micro`	1	1	低	使用 `mq.t2.micro` 实例类型进行 Amazon MQ 的基本评估。此实例类型（仅限单实例代理）符合 AWS 免费套餐条件。

实例类型	vCPU	内存 (GiB)	网络性能	注意
				Note 对 mq.t2.micro 实例类型的使用受 CPU 积分和基准性能的限制 – 能够突增至基准性能水平以上（有关更多信息，请参阅 CpuCreditBalance 指标）。如果您的应用程序需要固定性能，请考虑使用 mq.m5.large 实例类型。

实例类型	vCPU	内存 (GiB)	网络性能	注意
mq.t3.micro	2	1	低	使用 mq.t3.micro 实例类型进行 Amazon MQ 的基本评估。此实例类型（仅限单实例代理）符合AWS 免费套餐的资格。
mq.m4.large	2	8	中	使用 mq.m4.large 实例类型以实现与现有代理部署的兼容性。我们建议为新代理使用 mq.m5.* 实例。
mq.m5.large	2	8	高	将 mq.m5.large 实例用于常规开发、测试和生产工作负载。

实例类型	vCPU	内存 (GiB)	网络性能	注意
mq.m5.xlarge	4	16	高	将 mq.m5.xlarge 、mq.m5.2xlarge 和 mq.m5.4xlarge 实例类型用于需要高吞吐量的常规开发、测试和生产工作负载。 **Note** 当您的系统使用持久性消息时，其吞吐量取决于消息的使用速度。如果消息不会立即使用，则将较大实例类型与持久性消息结合使用可能不会提高系统吞吐量。在这种情况下，建议您将 concurren
mq.m5.2xlarge	8	32	高	
mq.m5.4xlarge	16	64	高	

实例类型	vCPU	内存 (GiB)	网络性能	注意
				tStoreAndDispatchQueues 属性设置为 false。有关更多信息，请参阅 对具有慢速使用者的队列禁用并发存储和分派。

有关吞吐量注意事项的更多信息，请参阅选择正确的代理实例类型以实现最佳吞吐量。

配置

配置中包含您的 ActiveMQ 代理的所有设置（采用 XML 格式，类似于 ActiveMQ 的 `activemq.xml` 文件）。您可以先创建配置，然后创建代理。之后，您可以将配置应用于一个或多个代理。

> **Important**
>
> 对配置进行更改不会立即将更改应用到代理。要应用更改，必须等待下一维护时段或者重启代理。有关更多信息，请参阅 Amazon MQ 代理配置生命周期。
> 目前，您无法删除配置。

有关创建、编辑和管理配置的信息，请参阅以下内容：

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

要跟踪您对配置所做的更改，可以创建配置版本。有关更多信息，请参阅 Creating and applying broker configurations。

Attributes

代理配置具有几个属性，例如：

- 名称 (`MyConfiguration`)
- ID (`c-1234a5b6-78cd-901e-2fgh-3i45j6k178l9`)
- Amazon Resource Name（ARN）(`arn:aws:mq:us-east-2:123456789012:configuration:c-1234a5b6-78cd-901e-2fgh-3i45j6k178l9`)

有关配置属性的完整列表，请参阅《Amazon MQ REST API 参考》中的以下内容：

- REST 操作 ID：Configuration
- REST 操作 ID：Configurations

有关配置修订属性的完整列表，请参阅以下内容：

- REST 操作 ID：Configuration Revision
- REST 操作 ID：Configuration Revisions

用户

ActiveMQ 用户是能够访问 ActiveMQ 代理的队列和主题的人或应用程序。您可以将用户配置为具有特定权限。例如，您可以允许某些用户访问 ActiveMQ Web 控制台。

组是一个语义标签。您可以为用户分配组，并为组配置发送、接收和管理特定队列和主题的权限。

> **Important**
>
> 对用户进行更改不会立即将更改应用于用户。要应用更改，必须等待下一维护时段或者重启代理。有关更多信息，请参阅 Amazon MQ 代理配置生命周期。

有关用户和组的信息，请参阅 Apache ActiveMQ 文档中的以下部分：

- 授权

- 授权示例

有关创建、编辑和删除 ActiveMQ 用户的信息，请参阅以下内容：

- 创建和管理 ActiveMQ 代理用户
- 用户

Attributes

有关用户属性的完整列表，请参阅《Amazon MQ REST API 参考》中的以下内容：

- REST 操作 ID：User
- REST 操作 ID：Users

Storage

Amazon MQ for ActiveMQ 支持 Amazon Elastic File System（EFS）和 Amazon Elastic Block Store（EBS）。默认情况下，ActiveMQ 代理使用 Amazon EFS 进行代理存储。要利用跨多个可用区的高持久性和复制功能，请使用 Amazon EFS。要利用低延迟和高吞吐量，请使用 Amazon EBS。

> **Important**
>
> - 您只能将 Amazon EBS 与 `mq.m5` 代理实例类型系列配合使用。
> - 尽管您可以更改代理实例类型，但在创建代理之后无法更改代理存储类型。
> - Amazon EBS 在单个可用区内复制数据，但不支持 ActiveMQ 主动/备用部署模式。

存储类型之间的差异

下表简要概述了 ActiveMQ 代理的内存、Amazon EFS 和 Amazon EBS 存储类型之间的差异。

存储类型	持久性	示例使用案例	每个创建器每秒排队消息的近似最大数量（1KB 消息）	复制
内存中	非持久性	• 股票报价	5000	无

存储类型	持久性	示例使用案例	每个创建器每秒排队消息的近似最大数量（1KB 消息）	复制
		• 位置数据更新 • 频繁更改的数据		
Amazon EBS	持续的	• 大量文本 • 订单处理	500	单个可用区 (AZ) 内的多个副本
Amazon EFS	持续的	金融交易	80	跨多个可用区的多个副本

内存中消息存储提供最低的延迟和最高的吞吐量。但是，在实例替换或代理重新启动期间，消息会丢失。

Amazon EFS 设计为提供高持久性，可跨多个可用区进行复制，以防止因任何单个组件故障或影响某个可用区可用性的问题而导致数据丢失。Amazon EBS 可针对吞吐量进行优化，并可在单个可用区中跨多个服务器进行复制。

代理架构

Amazon MQ for ActiveMQ 代理可作为单实例代理或主动/备用代理进行创建。对于这两种部署模式，Amazon MQ 通过冗余存储其数据来提供高持久性。

> **Note**
>
> Amazon MQ 使用 Apache KahaDB 作为其数据存储。不支持其他数据存储，如 JDBC 和 LevelDB。

您可以访问您的代理，方法是使用 ActiveMQ 支持的任何编程语言并通过为以下协议明确启用 TLS：

- AMQP
- MQTT
- MQTT 结束了 WebSocket

- OpenWire
- STOMP
- 大吃一惊 WebSocket

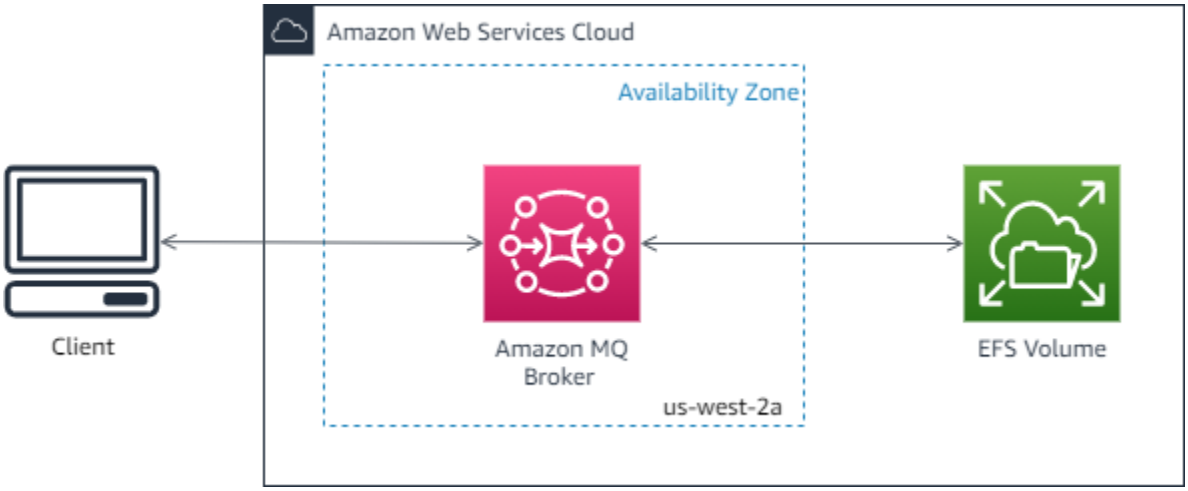
主题

- Amazon MQ 单实例代理
- 用于实现高可用性的 Amazon MQ 主动/备用代理
- Amazon MQ 代理网络

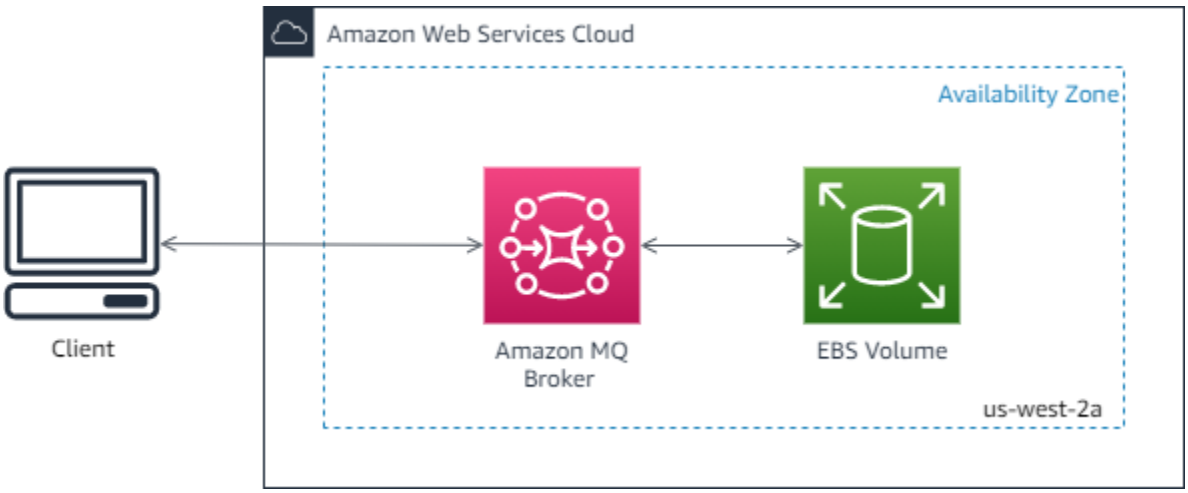
Amazon MQ 单实例代理

单实例代理由一个可用区中的一个代理组成。代理与您的应用程序以及 Amazon EBS 或 Amazon EFS 存储卷进行通信。Amazon EFS 存储卷旨在通过跨多个可用区（AZ）冗余存储数据来提供最高级别的持久性和可用性。Amazon EBS 提供针对低延迟和高吞吐量进行了优化的块级存储。有关存储选项的更多信息，请参阅Storage。

下图说明使用 Amazon EFS 存储的单实例代理跨多个可用区进行复制。



下图说明使用 Amazon EBS 存储的单实例代理在单个可用区中跨多个服务器进行复制。



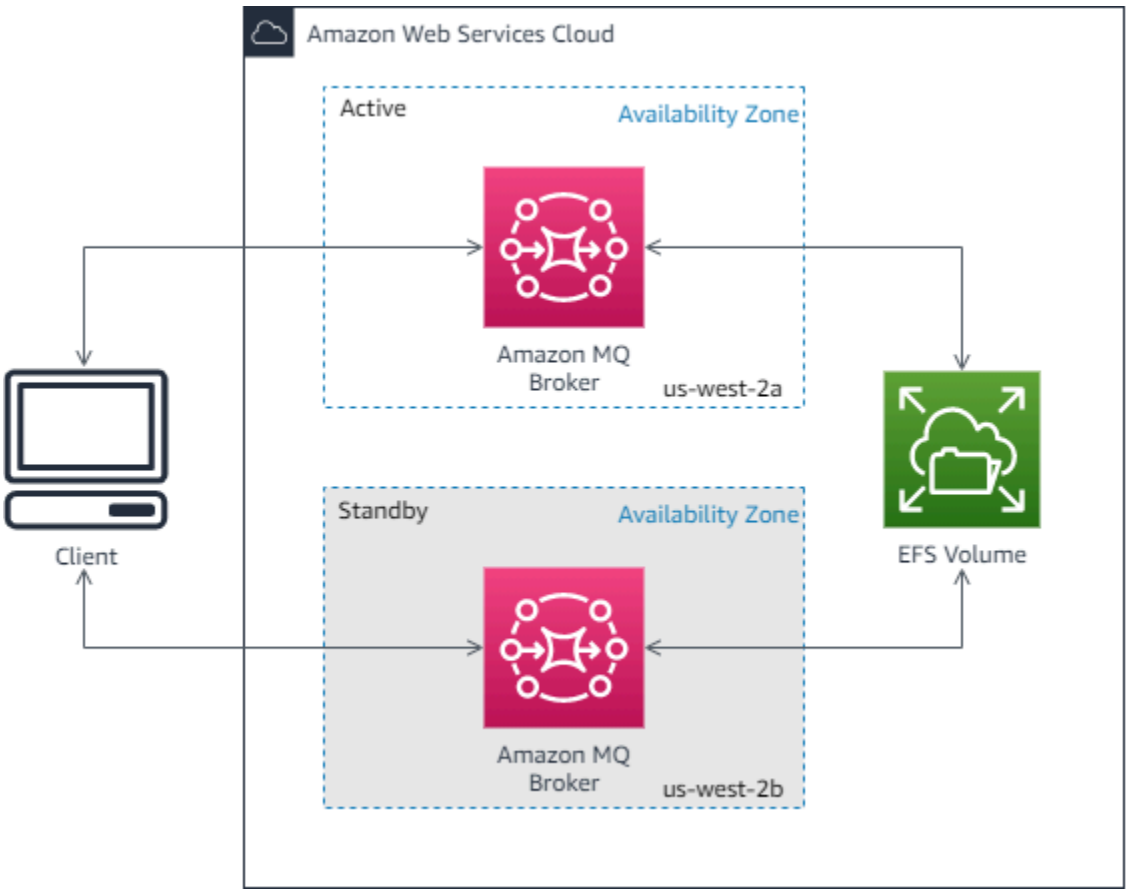
用于实现高可用性的 Amazon MQ 主动/备用代理

主动/备用代理由两个不同可用区中的两个代理组成，配置为冗余对。这些代理与您的应用程序以及 Amazon EFS 进行同步通信。Amazon EFS 存储卷旨在通过跨多个可用区（AZ）冗余存储数据来提供最高级别的持久性和可用性。有关更多信息，请参阅 Storage。

通常，任何时候都只有一个代理实例处于主动状态，其他代理实例则处于备用状态。如果其中一个代理实例出现故障或正在进行维护，则 Amazon MQ 需要花费一段时间才能使非活动实例停止服务。这允许运行状况良好的备用实例处于活动状态并开始接受传入通信。当您重启代理时，故障转移仅需几秒钟。

对于主动/备用代理，Amazon MQ 提供两个 ActiveMQ Web 控制台 URL，但每次只有一个 URL 处于活动状态。同样，Amazon MQ 为每个线级协议提供两个终端节点，但每次每对中只有一个终端节点处于活动状态。`-1` 和 `-2` 后缀表示冗余对。对于线级协议终端节点，您可以允许应用程序使用故障转移传输连接到任一终端节点。

下图说明使用 Amazon EFS 存储的主动/备用代理跨多个可用区进行复制。



Amazon MQ 代理网络

Amazon MQ 支持 ActiveMQ 代理网络功能。

代理网络由多个同时活动的单实例代理或主动/备用代理组成。您可以根据应用程序的需求（例如高可用性和可扩展性），采用各种拓扑（例如，集中器、中心辐射型、树形或网格）配置代理网络。例如，中心辐射型代理网络可以提高弹性，并在无法访问某个代理时保留消息。具有集中器拓扑的代理网络可以从接受传入消息的大量代理处收集消息，并将消息集中到更中心的代理，以更好地处理许多传入消息负载。

有关教程和详细配置信息，请参阅以下内容：

- Creating and Configuring a Network of Brokers
- 正确配置您的代理网络
- `networkConnector`
- `##ConnectionStart##`
- ActiveMQ 文档中的代理网络

以下是使用代理网络的好处：

- 创建代理网络能让您通过添加代理实例来增加聚合吞吐量和最大创建者与使用者连接计数。
- 您可以通过允许创建者和使用者了解多个活动代理实例来确保更好的可用性。这能让他们在当前连接到的实例变得不可用时重新连接到新实例。
- 因为创建者和使用者可以立即重新连接到代理网络中的另一个节点，并且因为不需要等待备用代理实例被提升，所以代理网络中的客户端重新连接比主动/备用代理更快，更容易实现高可用性。

主题

代理网络的工作原理是什么？

Amazon MQ 以多种方式支持 ActiveMQ 代理网络功能。首先，您可以编辑每个代理配置中的参数以创建代理网络，就像使用本机 ActiveMQ 一样。其次，Amazon MQ 具有使用 AWS CloudFormation 自动创建代理网络的示例蓝图。您可以直接从 Amazon MQ 控制台部署这些示例蓝图，也可以编辑相关的 AWS CloudFormation 模板以创建自己的拓扑和配置。

通过使用网络连接器将一个代理连接到另一个代理来建立代理网络。建立连接后，这些代理提供消息转发。例如，如果 Broker1 建立到 Broker2 的网络连接器，则 Broker1 上的消息将被转发到 Broker2（如果该代理上有使用者可用于队列或主题）。如果网络连接器配置为 `duplex`，则消息也会从 Broker2 转发到 Broker1。网络连接器是在代理配置中配置的。请参阅配置。例如，以下是代理配置中的示例 `networkConnector` 条目：

```
<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>
```

代理网络确保消息从一个代理实例流向另一个代理实例，从而仅将消息转发给具有相应使用者的代理实例。为助益网络中彼此相邻的代理实例，ActiveMQ 向建议主题发送有关创建者和使用者连接到网络和从网络断开的消息。当代理实例接收有关从特定目标中使用的使用方的信息时，代理实例会开始转发消息。有关更多信息，请参阅 ActiveMQ 文档中的建议主题。

代理网络如何处理凭据？

要使代理 A 连接到网络中的代理 B，代理 A 必须使用有效的凭据，就像任何其他创建者或使用者一样。您不必在代理 A 的 `<networkConnector>` 配置中提供密码，而必须首先在代理 A 上创建一个与代理 B 上的另一个用户具有相同值的用户（这些用户是共享相同用户名和密码值的独立且唯一的用户）。当您在 `<networkConnector>` 配置中指定 `userName` 属性时，Amazon MQ 将在运行时自动添加密码。

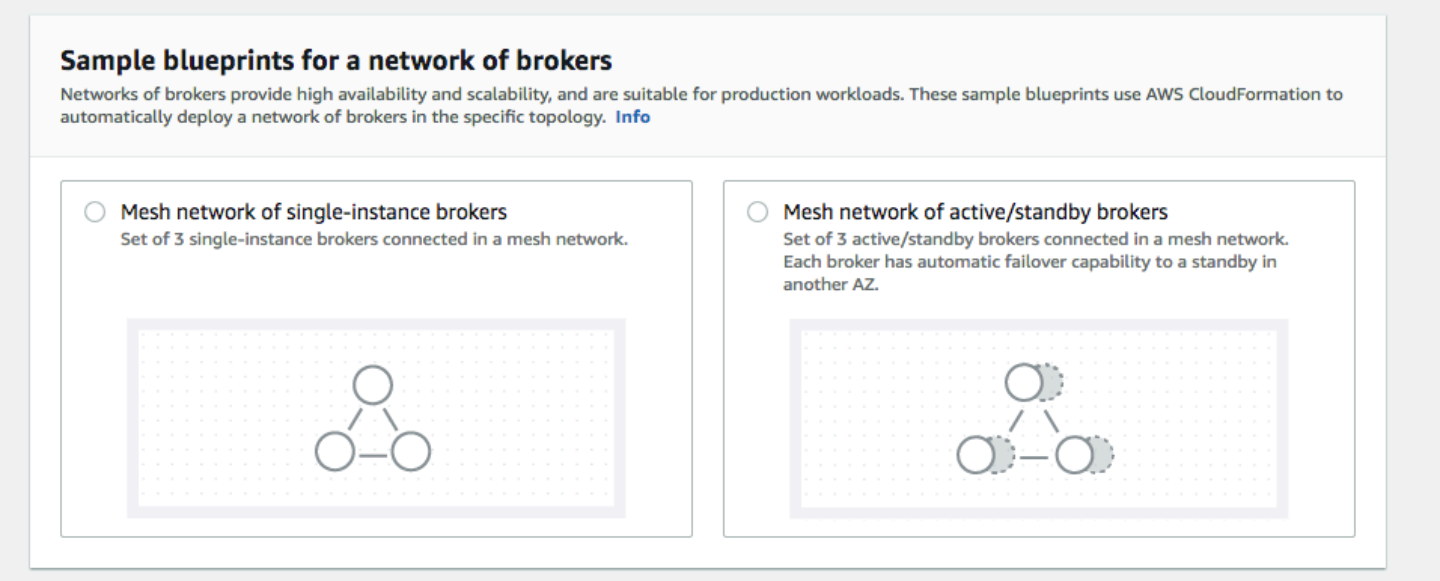
> **Important**
>
> 请勿为 `<networkConnector>` 指定 `password` 属性。我们不建议在代理配置文件中存储明文密码，因为这会使密码在 Amazon MQ 控制台中可见。有关更多信息，请参阅Configure Network Connectors for Your Broker。

代理必须位于相同的 VPC 或对等的 VPC 中。有关更多信息，请参阅Creating and Configuring a Network of Brokers教程中的先决条件。

示例蓝图

要开始使用代理网络，Amazon MQ 提供示例蓝图。这些示例蓝图使用 AWS CloudFormation 创建代理网络部署，以及所有相关资源。可用的两个示例蓝图是：

1. 单实例代理的网状网络
2. 主动/备用代理的网状网络



从 Create Broker (创建代理) 页面中，选择其中一个示例蓝图，然后选择 Next (下一步)。创建资源后，在 Amazon MQ 控制台中查看生成的代理及其配置。

通过在代理配置中创建代理并配置不同的 `networkConnector` 元素，您可以在许多不同的拓扑中创建代理网络。有关配置网络代理的更多信息，请参阅 ActiveMQ 文档中的代理网络。

代理网络拓扑

通过部署代理，然后在其配置中配置 `networkConnector` 条目，您可以构建使用不同网络拓扑的代理网络。网络连接器在连接的代理之间提供按需消息转发。可以将连接配置为双工，这样消息在代理之间双向转发，也可以配置为不是双工，这样转发仅从一个代理传播到另一个代理。例如，如果我们在 Broker1 和 Broker2 之间有双工连接，则消息将会在二者之间彼此转发（如果有使用者）。

有了双工网络连接器，消息将从每个代理转发到另一个代理。这些是按需转发的：如果 Broker2 上有用户想要获取 Broker1 上的消息，可转发该消息。同样，如果 Broker1 上的用户想要获取 Broker2 上的消息，也可转发该消息。

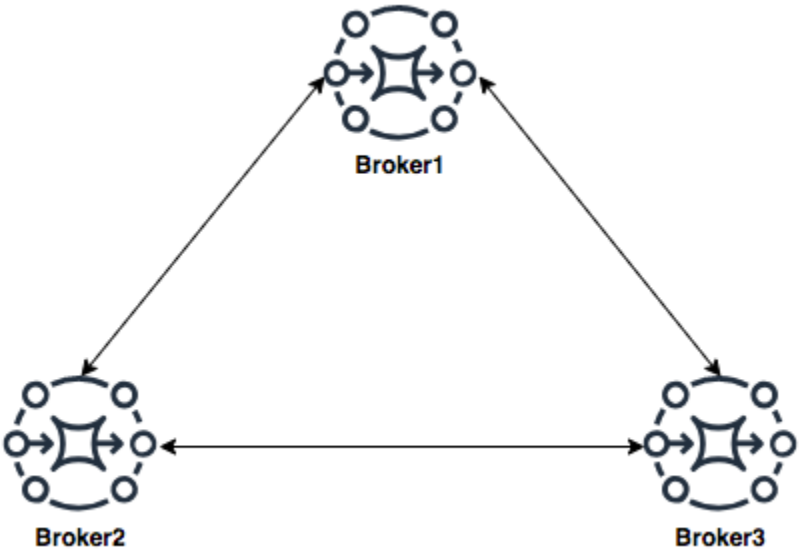
对于非双工连接，消息仅从一个代理转发到另一个代理。在本示例中，如果 Broker2 上的用户想要获取 Broker1 上的消息，可转发该消息。但是，消息不会从 Broker2 转发到 Broker1。

使用双工和非双工网络连接器，可以在任意数量的网络拓扑中构建代理网络。

> **Note**
>
> 在每个网络拓扑示例中，`networkConnector` 元素都会引用它们连接到的代理的终端节点。将 `uri` 属性中的代理终端节点条目替换为代理的终端节点。请参阅Listing brokers and viewing broker details。

网状拓扑

网状拓扑提供了多个彼此连接的代理。这个简单的示例连接了三个单实例代理，但您可以将更多代理配置为网格。



此拓扑以及一个包含主动/备用代理对网格的拓扑，可以使用 Amazon MQ 控制台中的示例蓝图创建。您可以创建这些示例蓝图部署以查看正在工作的代理网络，并查看它们的配置方式。

您可以通过向 Broker1 添加网络连接器来配置这样的三个代理网状网络，该连接器与 Broker2 和 Broker3 都建立双工连接 ，在 Broker2 和 Broker3 之间建立单个双工连接。

Broker1 的网络连接器：

```
<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>
```

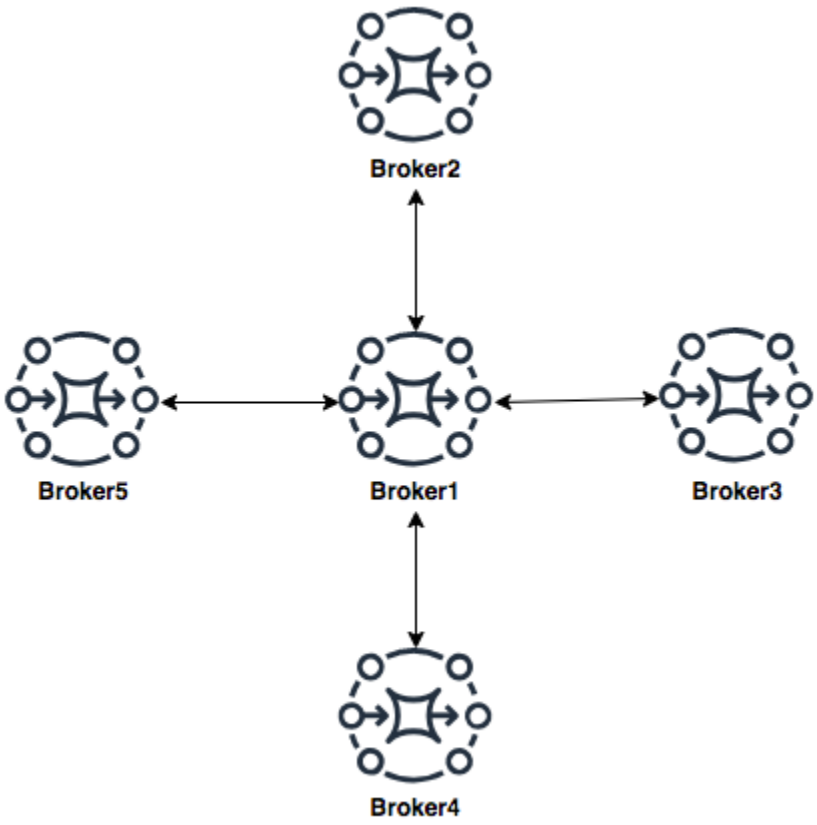
Broker2 的网络连接器：

```
<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>
```

通过将上述连接器添加到 Broker1 和 Broker2 的配置中，您可以在这三个代理之间创建一个网格，从而在所有代理之间按需转发消息。有关更多信息，请参阅Amazon MQ Broker Configuration Parameters。

中心辐射型拓扑

在中心辐射型拓扑中，如果轮辐上的任何代理中断，则会保留消息。消息对全程转发，而且只有中心的 Broker1 对网络的运行至关重要。



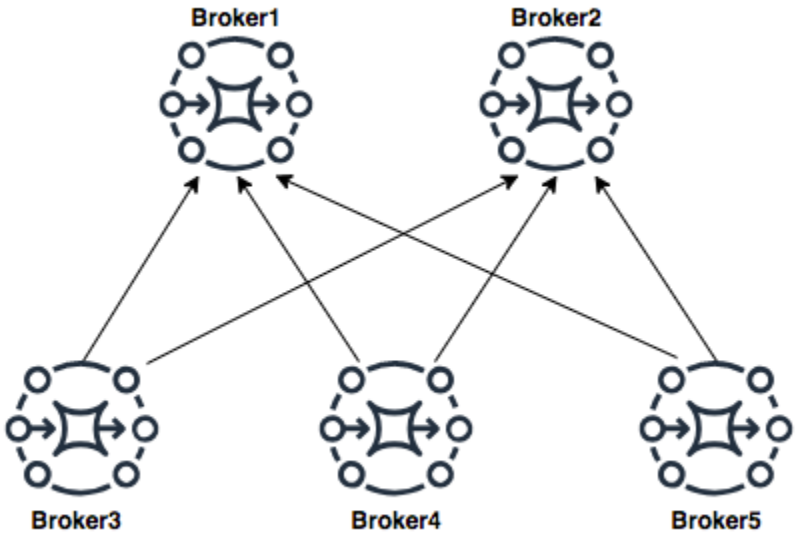
在本示例中，要配置代理的中心辐射网络，您可以在 Broker1 的配置中向辐条上的每个代理添加networkConnector。

```
<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>
```

集中器拓扑

在此示例拓扑中，底部的三个代理可以处理大量连接，这些消息集中到 Broker1 和 Broker2。每个其他代理都与更中心的代理建立非双工连接。要扩展此拓扑的容量，您可以添加接收消息的附加代理并将这些消息集中在 Broker1 和 Broker2 中。



要配置此拓扑，底部的每个代理都将包含一个网络连接器，用于连接其将消息集中到的每个代理。

Broker3 的网络连接器：

```
<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 的网络连接器：

```
<networkConnectors>
```

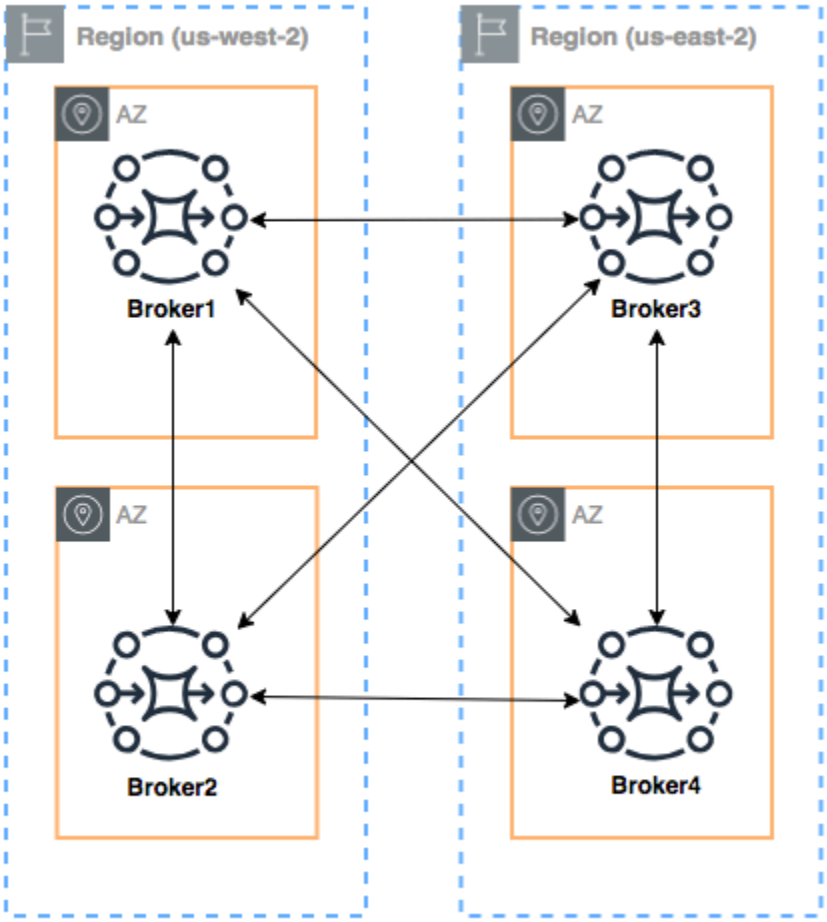
```
    <networkConnector name="4_to_1" userName="myCommonUser" duplex="false"
        uri="static:(ssl://b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-
east-2.amazonaws.com:61617)"/>
    <networkConnector name="4_to_2" userName="myCommonUser" duplex="false"
        uri="static:(ssl://b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-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

Broker5 的网络连接器：

```
<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>
```

跨区域

要配置跨 AWS 区域的代理网络，请在这些区域中部署代理，并将网络连接器配置到这些代理的终端节点。



要配置像此示例一样的代理网络，您可以将 networkConnectors 条目添加到 Broker1 和 Broker4 的配置中，以引用这些代理的线级终端节点。

Broker1 的网络连接器：

```
<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 的网络连接器：

```
<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 的网络连接器：

```
<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>
```

借助传输连接器进行的动态故障转移

除了配置 networkConnector 元素，您还可以配置您的代理 transportConnector 选项以启用动态故障转移，以及在从网络中添加或删除代理后重新平衡连接。

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

在本示例中，updateClusterClients 和 rebalanceClusterClients 均设置为 true。在这种情况下，系统会向客户端提供网络中的代理的列表，而且客户端会在有新代理加入时请求这些代理重新平衡。

可用选项:

- updateClusterClients: 向客户端传递有关代理拓扑网络中的更改的信息。
- rebalanceClusterClients: 导致客户端在有新代理添加到代理网络时跨这些代理重新平衡。
- updateClusterClientsOnRemove: 在有代理离开代理网络时，更新客户端的拓扑信息。

当设置 updateClusterClients 为 true 时，客户端可以配置为连接到网络代理中的单个代理。

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

当有新代理连接时，它将收到在网络中的所有代理的 URI 列表。如果连接到代理失败，它可以动态切换到在其连接时所提供的代理之一。

有关故障转移的更多信息，请参阅 Active MQ 文档中的用于故障转移的代理端选项。

Amazon MQ for ActiveMQ 代理配置

配置中包含您的 ActiveMQ 代理的所有设置（采用 XML 格式，类似于 ActiveMQ 的 activemq.xml 文件）。您可以先创建配置，然后创建代理。之后，您可以将配置应用于一个或多个代理。

主题

- 使用 Spring XML 配置文件
- 创建、编辑和应用 ActiveMQ 代理配置
- Amazon MQ 配置中允许的元素
- Amazon MQ 配置中允许的元素及其属性
- Amazon MQ 配置中允许的元素、子集合元素及其子元素

使用 Spring XML 配置文件

使用 Spring XML 文件来配置 ActiveMQ 代理。您可以配置您的 ActiveMQ 代理的许多方面，如预定义目标、目标策略、授权策略和插件。Amazon MQ 控制其中一些配置元素，如网络传输和存储。目前不支持其他配置选项，如创建代理的网络。

在 Amazon MQ XML 架构中指定了全套受支持的配置选项：您可以使用以下链接下载受支持架构的 zip 文件。

- 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

您可以使用这些架构来验证和清理您的配置文件。Amazon MQ 还允许您通过上传 XML 文件来提供配置。在您上传 XML 文件时，Amazon MQ 会自动根据架构来清理和删除无效的和禁止的配置参数。

> **Note**
>
> 您只能使用属性的静态值。Amazon MQ 会从您的配置中清理包含 Spring 表达式、变量和元素引用的元素和属性。

创建、编辑和应用 ActiveMQ 代理配置

配置中包含您的 ActiveMQ 代理的所有设置（采用 XML 格式，类似于 ActiveMQ 的 `activemq.xml` 文件）。您可以先创建配置，然后创建代理。之后，您可以将配置应用于一个或多个代理。您可以立即应用或在维护时段内应用配置。

有关更多信息，请参阅下列内容：

- 配置
- Amazon MQ 代理配置生命周期
- Amazon MQ Broker Configuration Parameters

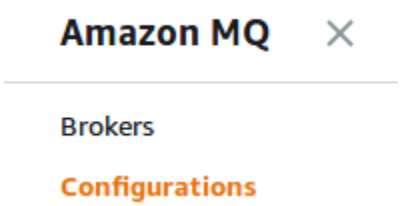
以下示例演示如何使用 AWS Management Console创建和应用 Amazon MQ 代理配置。

主题

- 创建新的配置
- 创建新的配置修订
- 将配置修订应用到代理
- 编辑配置修订

创建新的配置

1. 登录 Amazon MQ 控制台。
2. 在左侧，展开导航面板，然后选择 Configurations (配置)。



3. 在 Configurations (配置) 页面上，选择 Create configuration (创建配置)。

4. 在 Create configuration（创建配置）页面上的 Details（详细信息）部分中，输入 Configuration name（配置名称）（例如 `MyConfiguration`）并选择 Broker engine（代理引擎）版本。

> **Note**
>
> 要了解有关 Amazon MQ for ActiveMQ 支持的 ActiveMQ 引擎版本的更多信息，请参阅the section called “版本管理”。

5. 选择创建配置。

创建新的配置修订

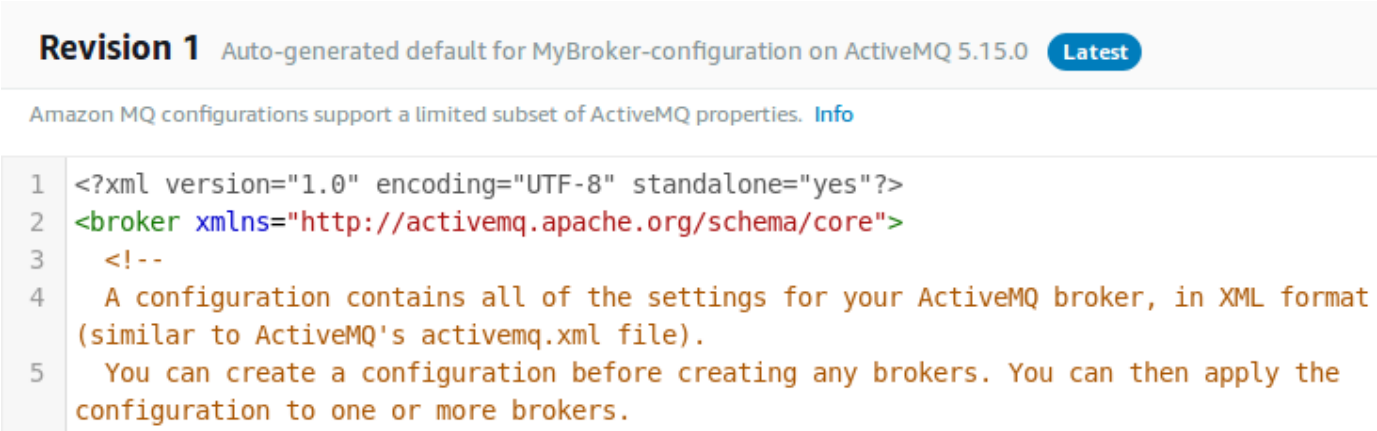
1. 从配置列表中选择***MyConfiguration***。

> **Note**
>
> 始终会在 Amazon MQ 创建配置时为您创建第一个配置修订。

在该***MyConfiguration***页面上，将显示您的新配置修订版使用的代理引擎类型和版本（例如 Apache ActiveMQ 5.15.16）。

2. 在 Configuration details 选项卡上，会显示配置修订号、描述和 XML 格式的代理配置。

> **Note**
>
> 编辑当前配置会创建一个新的配置修订。



3. 选择 Edit configuration (编辑配置) 并对 XML 配置进行更改。
4. 选择保存。

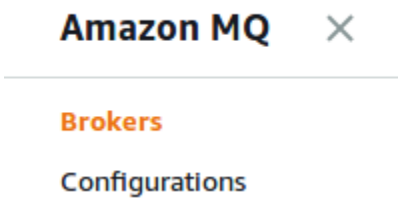
   Save revision (保存修订) 对话框出现。
5. （可选）类型 `A description of the changes in this revision`。
6. 选择保存。

   将会保存配置的新修订。

> **Important**
>
> Amazon MQ 控制台会自动根据架构来清理无效和禁止的配置参数。有关更多信息和允许的 XML 参数的完整列表，请参阅Amazon MQ Broker Configuration Parameters。
> 对配置进行更改不会立即将更改应用到代理。要应用更改，必须等待下一维护时段或者重启代理。有关更多信息，请参阅 Amazon MQ 代理配置生命周期。
> 目前，您无法删除配置。

将配置修订应用到代理

1. 在左侧，展开导航面板，然后选择 Brokers (代理)。



2. 从经纪人列表中，选择您的经纪商（例如 MyBroker），然后选择编辑。
3. 在 “编辑 ***MyBroker***” 页面的 “配置” 部分，选择配置和修订版，然后选择 “计划修改”。
4. 在 Schedule broker modifications (计划代理修改) 部分中，选择是在 During the next scheduled maintenance window (下一个计划维护时段期间) 还是 Immediately (立即) 应用修改。

> **Important**
>
> 您的代理将在重启时脱机。

5. 选择 应用。

您的配置修订将在指定的时间应用到您的代理。

编辑配置修订

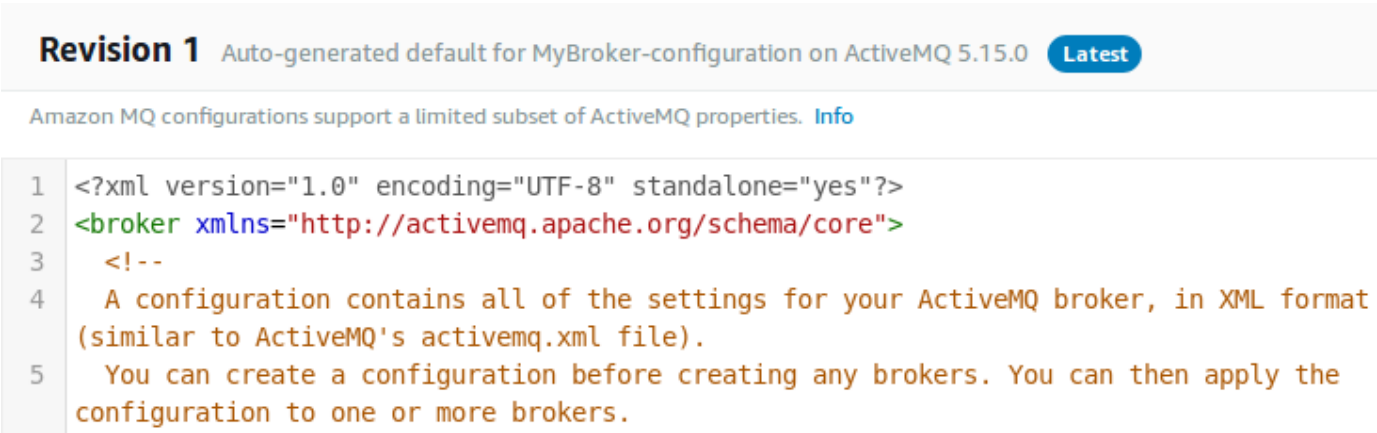
1. 登录 Amazon MQ 控制台。
2. 从经纪人列表中，选择您的经纪商（例如 MyBroker），然后选择编辑。
3. 在***MyBroker***页面上，选择 “编辑”。
4. 在 “编辑 ***MyBroker***” 页面的 “配置” 部分，选择一个配置和一个修订版，然后选择编辑。

> **Note**
>
> 除非您在创建代理时选择配置，否则会在 Amazon MQ 创建代理时为您创建第一个配置修订。

在该***MyBroker***页面上，将显示配置使用的代理引擎类型和版本（例如 Apache ActiveMQ 5.15.8）。

5. 在 Configuration details 选项卡上，会显示配置修订号、描述和 XML 格式的代理配置。

> **Note**
>
> 编辑当前配置会创建一个新的配置修订。



6. 选择 Edit configuration (编辑配置) 并对 XML 配置进行更改。
7. 选择保存。

Save revision (保存修订) 对话框出现。

8. （可选）类型 `A description of the changes in this revision`。
9. 选择保存。

将会保存配置的新修订。

> **Important**
>
> Amazon MQ 控制台会自动根据架构来清理无效和禁止的配置参数。有关更多信息和允许的 XML 参数的完整列表，请参阅Amazon MQ Broker Configuration Parameters。
> 对配置进行更改不会立即将更改应用到代理。要应用更改，必须等待下一维护时段或者重启代理。有关更多信息，请参阅 Amazon MQ 代理配置生命周期。
> 目前，您无法删除配置。

Amazon MQ 配置中允许的元素

下面是 Amazon MQ 配置中允许的元素的详细列表。有关更多信息，请参阅 Apache ActiveMQ 文档中的 XML 配置。

元素
`abortSlowAckConsumerStrategy` (属性)
`abortSlowConsumerStrategy` (属性)
`authorizationEntry` (属性)
`authorizationMap` (子集合元素)
`authorizationPlugin` (子集合元素)
`broker` (属性 | 子集合元素)
`cachedMessageGroupMapFactory` (属性)
`compositeQueue` (属性 | 子集合元素)
`compositeTopic` (属性 | 子集合元素)

元素
constantPendingMessageLimitStrategy (属性)
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 | 子集合元素)
policyMap (子集合元素)
prefetchRatePendingMessageLimitStrategy (属性)

元素
priorityDispatchPolicy
priorityNetworkDispatchPolicy
queryBasedSubscriptionRecoveryPolicy (属性)
queue (属性)
redeliveryPlugin (属性 | 子集合元素)
redeliveryPolicy (属性)
redeliveryPolicyMap (子集合元素)
retainedMessageSubscriptionRecoveryPolicy (子集合元素)
roundRobinDispatchPolicy
sharedDeadLetterStrategy (属性 | 子集合元素)
simpleDispatchPolicy
simpleMessageGroupMapFactory
statisticsBrokerPlugin
storeCursor
storeDurableSubscriberCursor (属性)
strictOrderDispatchPolicy
tempDestinationAuthorizationEntry (属性)
tempQueue (属性)
tempTopic (属性)
timedSubscriptionRecoveryPolicy (属性)

元素
timeStampingBrokerPlugin (属性)
topic (属性)
transportConnector (属性)
uniquePropertyMessageEvictionStrategy (属性)
virtualDestinationInterceptor (子集合元素)
virtualTopic (属性)
vmCursor
vmDurableCursor
vmQueueCursor

Amazon MQ 配置中允许的元素及其属性

下面是 Amazon MQ 配置中允许的元素及其属性的详细列表。有关更多信息，请参阅 Apache ActiveMQ 文档中的 XML 配置。

元素	属性
abortSlowAckConsumerStrategy	abortConnection
	checkPeriod
	ignoreIdleConsumers
	ignoreNetworkConsumers
	maxSlowCount
	maxSlowDuration
	maxTimeSinceLastAck

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

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

元素	属性
	timeBeforePurgeTempDestinations
	useAuthenticatedPrincipalFo rJMSXUserID
	useMirroredQueues
	useTempMirroredQueues
	useVirtualDestSubs
	useVirtualDestSubsOnCreation
	useVirtualTopics
cachedMessageGroupMapFactory	cacheSize
compositeQueue	concurrentSend
	copyMessage
	forwardOnly
	name
	sendWhenNotMatched
compositeTopic	concurrentSend
	copyMessage
	forwardOnly
	name
	sendWhenNotMatched
有条件的 NetworkBridge FilterFactory	rateDuration
	rateLimit

元素	属性
	replayDelay
	replayWhenNoConsumers
	selectorAware 在以下版本中受支持： Apache ActiveMQ 5.16.x
constantPendingMessageLimitStrategy	limit
discarding	deadLetterQueue
	enableAudit
	expiration
	maxAuditDepth
	maxProducersToAudit
	processExpired
	processNonPersistent
discardingDLQBrokerPlugin	dropAll
	dropOnly
	dropTemporaryQueues
	dropTemporaryTopics
	reportInterval
filteredDestination	queue

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

元素	属性
	cacheSize
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

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

元素	属性
	persistJMSRedelivered
	prioritizedMessages
	producerFlowControl
	queue
	queueBrowserPrefetch
	queuePrefetch
	reduceMemoryFootprint
	sendAdvisoryIfNoConsumers
	sendFailIfNoSpace
	sendFailIfNoSpaceAfterTimeout 在以下版本中受支持： Apache ActiveMQ 5.16.4 及更高版本
	sendDuplicateFromStoreToDLQ
	storeUsageHighWaterMark
	strictOrderDispatch
	tempQueue
	tempTopic
	timeBeforeDispatchStarts
	topic
	topicPrefetch

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

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

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

元素	属性
	name
	postfix
	prefix
	selectorAware
	setOriginalDestination
	transactedSend

Amazon MQ 父元素属性

下面是父元素属性的详细说明。有关更多信息，请参阅 Apache ActiveMQ 文档中的 XML 配置。

主题

- 代理

代理

broker 是一个父集合元素。

Attributes

网络ConnectionStart异步

要缓解网络延迟并允许其他网络及时启动，请使用 <networkConnectionStartAsync> 标签。该标签指示代理使用执行程序并行启动网络连接（与代理启动异步）。

默认值：false

示例配置

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

Amazon MQ 配置中允许的元素、子集合元素及其子元素

下面是 Amazon MQ 配置中允许的元素、子集合元素及其子元素的详细列表。有关更多信息，请参阅 Apache ActiveMQ 文档中的 XML 配置。

元素	子集合元素	子元素
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
	persistenceAdapter	kahaDB

元素	子集合元素	子元素
	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
compositeTopic	forwardTo	queue

元素	子集合元素	子元素
		tempQueue
		tempTopic
		topic
		filteredDestination
policyEntry	deadLetterStrategy	discarding
		individualDeadLetterStrategy
		sharedDeadLetterStrategy
	destination	queue
		tempQueue
		tempTopic
		topic
	dispatchPolicy	priorityDispatchPolicy
		priorityNetworkDispatchPolicy
		roundRobinDispatchPolicy
		simpleDispatchPolicy
		strictOrderDispatchPolicy

元素	子集合元素	子元素
		clientIdFilterDispatchPolicy
	messageEvictionStrategy	oldestMessageEvictionStrategy
		oldestMessageWithLowestPriorityEvictionStrategy
		uniquePropertyMessageEvictionStrategy
	messageGroupMapFactory	cachedMessageGroupMapFactory
		messageGroupHashBucketFactory
		simpleMessageGroupMapFactory
	pendingDurableSubscriberPolicy	fileDurableSubscriberCursor
		storeDurableSubscriberCursor
		vmDurableCursor
	pendingMessageLimitStrategy	constantPendingMessageLimitStrategy
		prefetchRatePendingMessageLimitStrategy
	pendingQueuePolicy	fileQueueCursor

元素	子集合元素	子元素
		storeCursor
		vmQueueCursor
	pendingSubscriberPolicy	fileCursor
		vmCursor
	slowConsumerStrategy	abortSlowAckConsumerStrategy
		abortSlowConsumerStrategy
	subscriptionRecoveryPolicy	fixedCountSubscriptionRecoveryPolicy
		fixedSizedSubscriptionRecoveryPolicy
		lastImageSubscriptionRecoveryPolicy
		noSubscriptionRecoveryPolicy
		queryBasedSubscriptionRecoveryPolicy
		retainedMessageSubscriptionRecoveryPolicy
timedSubscriptionRecoveryPolicy		
policyMap	defaultEntry	policyEntry

元素	子集合元素	子元素
	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

元素	子集合元素	子元素
virtualDestination Interceptor	virtualDestinations	compositeQueue
		compositeTopic
		virtualTopic

Amazon MQ 子元素属性

下面是子元素属性的详细说明。有关更多信息，请参阅 Apache ActiveMQ 文档中的 XML 配置。

主题

- authorizationEntry
- networkConnector
- kahaDB
- systemUsage

authorizationEntry

authorizationEntry 是 authorizationEntries 子集合元素的子项。

属性

admin|read|write

授予一组用户的权限。有关更多信息，请参阅始终配置授权映射。

如果您指定的授权映射不包含在 activemq-webconsole 组中，您无法使用 ActiveMQ Web 控制台，因为该组未获得授权向 Amazon MQ 代理发送消息或接收来自该代理的消息。

默认值：null

示例配置

```
<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 是 networkConnectors 子集合元素的子项。

主题

- 属性
- 示例配置

属性

conduitSubscriptions

指定代理网络中的网络连接是否将订阅同一目标的多个使用者视为一个使用者。例如，如果 conduitSubscriptions 设置为 true，并且两个使用者连接到代理 B 并从目标中使用，则代理 B 会通过与代理 A 的网络连接将订阅组合到单个逻辑订阅中，以便只有一个消息副本从代理 A 转发到代理 B。

> **Note**
>
> 将 conduitSubscriptions 设置为 true 可以减少冗余网络流量。但是，使用此属性可能会影响跨使用者的消息负载均衡，并可能在某些情况下导致不正确的行为（例如，对于 JMS 消息选择器或持久主题）。

默认值：true

duplex

指定代理网络中的连接是否用于生成和使用消息。例如，如果代理 A 在非双工模式下创建与代理 B 的连接，则消息只能从代理 A 转发到代理 B。但是，如果代理 A 创建到代理 B 的双工连接，则代理 B 可以将消息转发到代理 A 而无需配置 <networkConnector>。

默认值：`false`

name

代理网络中的网桥的名称。

默认值：`bridge`

uri

代理网络中两个代理（或多个代理）之一的有线级协议终端节点。

默认值：`null`

username

代理网络中的代理共有的用户名。

默认值：`null`

示例配置

> **Note**
>
> 当使用 `networkConnector` 来定义代理网络时，请勿包含代理共有的用户的密码。

有两个代理的代理网络

在此配置中，两个代理连接在代理网络中。网络连接器的名称是 `connector_1_to_2`，代理共有的用户名是 `myCommonUser`，连接是 `duplex`，OpenWire 终端节点 URI 以 `static:` 为前缀，表示代理之间的一对一连接。

```
<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>
```

有关更多信息，请参阅Configure Network Connectors for Your Broker。

有多个代理的代理网络

在此配置中，多个代理连接在代理网络中。网络连接器的名称是 connector_1_to_2，代理共有的用户名是 myCommonUser，连接是 duplex，逗号分隔的 OpenWire 终端节点 URI 列表以 masterslave: 为前缀，表示代理之间的故障转移连接。从代理到代理的故障转移不是随机的，重新连接尝试将无限期地继续。

```
<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**
>
> 我们建议您对网络代理使用 masterslave: 前缀。该前缀与更明确的 static:failover:()?randomize=false&maxReconnectAttempts=0 语法相同。

> **Note**
>
> 此 XML 配置不允许使用空格。

kahaDB

kahaDB 是 persistenceAdapter 子集合元素的子项。

属性

concurrentStoreAndDispatchQueues

指定是否对队列使用并发存储和分派。有关更多信息，请参阅对具有慢速使用者的队列禁用并发存储和分派。

默认值：true

cleanupOnStop

> 在以下版本中受支持：
>
> Apache ActiveMQ 15.16.x 及更高版本

如果停用，则不会在代理停止时进行垃圾回收和清理，从而加快关闭过程。在使用大型数据库或调度器数据库的情况下，提高速度非常有用。

默认值：`true`

journalDiskSyncInterval

在 `journalDiskSyncStrategy=periodic` 的情况下，执行磁盘同步的时间间隔 (毫秒)。有关详细信息，请参阅 Apache ActiveMQ kahaDB 文档。

默认值：`1000`

journalDiskSyncStrategy

> 在以下版本中受支持：
>
> Apache ActiveMQ 15.14.x 及更高版本

配置磁盘同步策略。有关详细信息，请参阅 Apache ActiveMQ kahaDB 文档。

默认值：`always`

> Note
>
> ActiveMQ 文档规定数据损失仅限于持续时间 `journalDiskSyncInterval`，其默认值为 1 秒。数据损失可以超过该间隔，但很难精确确定。请谨慎使用。

preallocationStrategy

配置在需要新日志文件时，代理尝试预分配日志文件的方式。有关详细信息，请参阅 Apache ActiveMQ kahaDB 文档。

默认值：sparse_file

示例配置

Example

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

systemUsage

systemUsage 是 systemUsage 子集合元素的子项。该元素控制代理在减缓创建器速度之前使用的最大空间量。有关更多信息，请参阅 Apache ActiveMQ 文档中的创建器流控制。

子元素

memoryUsage

memoryUsage 是 systemUsage 子元素的子项。该元素管理内存使用情况。使用 memoryUsage 可以跟踪正在使用的内存量，这样您可以高效地控制工作集的使用情况。有关更多信息，请参阅 Apache ActiveMQ 文档中的架构。

子元素

memoryUsage 是 memoryUsage 子元素的子项。

属性

percentOfJvmHeap

介于 0（含）与 70（含）之间的整数。

默认值：70

属性

sendFailIfNoSpace

设置在没有可用空间的情况下 send() 方法是否应失败。默认值为 false，该值会阻止 send() 方法，直至有空间可用。有关更多信息，请参阅 Apache ActiveMQ 文档中的架构。

默认值：`false`

sendFailIfNoSpaceAfterTimeout

默认值：`null`

示例配置

Example

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

管理 Amazon MQ for ActiveMQ 引擎版本

Apache ActiveMQ 根据语义版本控制规范将版本号整理为 `X.Y.Z`。在适用于 ActiveMQ 实现的 Amazon MQ 中，X表示主版本，代表次要版本，Y表示补丁版本号。Z如果主要版本号发生变化，Amazon MQ 会将版本更改视为主要版本更改。例如，从版本 5. 17 升级到 6. 0 被视为主要版本升级。如果只有次要版本号或补丁版本号发生更改，则版本更改被视为次要更改。例如，从版本 5 升级。 17 到 5。 18 被视为次要版本升级。

适用于 ActiveMQ 的亚马逊 MQ 建议所有经纪商使用支持的最新次要版本。有关如何升级您的代理引擎版本的说明，请参阅升级 Amazon MQ 代理引擎版本。

亚马逊 MQ 上支持的 ActiveMQ 引擎版本

Amazon MQ 版本支持日历会显示代理引擎版本何时终止支持。当某个版本的支持终止时，Amazon MQ 会自动将该版本上的所有代理升级到下一个支持的版本。在版本终止支持之前，Amazon MQ 会至少在 90 天内发出通知。

Apache ActiveMQ 版本	亚马逊 MQ 的支持已终止
ActiveMQ 5.17（推荐）	

Apache ActiveMQ 版本	亚马逊 MQ 的支持已终止
ActiveMQ 5.16	2024年11月15日
ActiveMQ 5.15	2024年9月16日

当您为 ActiveMQ 代理创建新的 Amazon MQ 时，您可以指定任何受支持的 ActiveMQ 引擎版本。如果您使用创建代理，Amazon MQ 会自动默认为最新的引擎版本号。 AWS Management Console 如果您使用 AWS CLI 或 Amazon MQ API 创建代理，则需要引擎版本号。如果不提供版本号，则操作会导致异常。要了解更多信息，请参阅《AWS CLI 命令参考》中的`create-broker`和《Amazon MQ REST API 参考》中的`CreateBroker`。

引擎版本升级

您可以随时手动将您的代理升级到下一个支持的主要版本、次要版本或补丁版本。当您开启自动次要版本升级时，Amazon MQ 将在维护时段内将您的代理升级到支持的最新补丁版本。

有关手动升级经纪商的更多信息，请参阅the section called “升级引擎版本”。

列出支持的引擎版本

您可以使用`describe-broker-instance-options` AWS CLI 命令列出所有支持的次要和主要引擎版本。

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

要按引擎和实例类型筛选结果，请使用`--engine-type`和`--host-instance-type`选项，如以下所示。

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

例如，要筛选 ActiveMQ 和 `mq.m5.large` 实例类型的结果，请将 *`engine-type`* 替换为 `ACTIVEMQ`，并将 *`instance-type`* 替换为 `mq.m5.large`。

将 Java Message Service（JMS）与 ActiveMQ 配合使用的有效示例

以下示例演示如何以编程方式使用 ActiveMQ：

- 示 OpenWire 例 Java 代码连接到代理、创建队列以及发送和接收消息。有关详细分解和说明，请参阅Connecting a Java application to your broker。
- MQTT 示例 Java 代码可连接到代理、创建主题并发送和接收消息。
- STOMP+WSS 示例 Java 代码可连接到代理、创建队列并发送和接收消息。

先决条件

启用 VPC 属性

要确保您的代理可以在您的 VPC 中访问，您必须启用 `enableDnsHostnames` 和 `enableDnsSupport` VPC 属性。有关更多信息，请参阅《Amazon VPC 用户指南》中的 VPC 中的 DNS Support。

启用入站连接

1. 登录 Amazon MQ 控制台。
2. 从经纪人列表中，选择您的经纪商的名称（例如 MyBroker）。
3. 在该***MyBroker***页面的 “连接” 部分，记下代理的 Web 控制台 URL 和线级协议的地址和端口。
4. 在 Details (详细信息) 部分的 Security and network (安全与网络) 下，选择您的安全组名称或 。

   此时将显示 EC2 Dashboard 的 Security Groups (安全组) 页面。
5. 从安全组列表中，选择您的安全组。
6. 在页面底部，选择 Inbound (入站)，然后选择 Edit (编辑)。
7. 在 Edit inbound rules (编辑入站规则) 对话框中，为希望公开访问的每个 URL 或终端节点添加规则（以下示例显示如何为代理 Web 控制台执行此操作。

   a. 选择添加规则。

   b. 对于 Type (类型)，选择 Custom TCP (自定义 TCP)。

   c. 对于 Port Range (端口范围)，键入 Web 控制台端口（`8162`）。

   d. 对于 Source (源)，选择 Custom (自定义)，然后键入您希望能够访问 Web 控制台的系统的 IP 地址（例如 `192.0.2.1`)。

   e. 选择保存。

   您的代理现在可以接受入站连接。

添加 Java 依赖项

OpenWire

将 activemq-client.jar 和 activemq-pool.jar 程序包添加到 Java 类路径中。以下示例说明了 Maven 项目的 pom.xml 文件中的这些依赖关系。

```
<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>
```

有关 activemq-client.jar 的更多信息，请参阅 Apache ActiveMQ 文档中的初始配置。

MQTT

将 org.eclipse.paho.client.mqttv3.jar 程序包添加到 Java 类路径中。以下示例说明了 Maven 项目的 pom.xml 文件中的这一依赖关系。

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

有关 org.eclipse.paho.client.mqttv3.jar 的更多信息，请参阅 Eclipse Paho Java 客户端。

STOMP+WSS

将以下程序包添加到了 Java 类路径：

- spring-messaging.jar

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

以下示例说明了 Maven 项目的 pom.xml 文件中的这些依赖关系。

```
<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>
```

有关更多信息，请参阅 Spring Framework 文档中的 STOMP 支持。

AmazonMQExample.java

> **Important**
>
> 在以下示例代码中，生产者和使用者在单个线程中运行。对于生产系统（或测试代理实例故障转移），请确保您的创建者和使用者在单独的主机或线程上运行。

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 教程

以下教程介绍如何创建和连接到 ActiveMQ 代理。要使用 ActiveMQ Java 示例代码，您必须安装 Java 标准版开发工具包并对代码进行一些更改。

主题

- 创建和配置 ActiveMQ 代理
- 创建和配置 Amazon MQ 代理网络
- 将 Java 应用程序连接到您的 Amazon MQ 代理
- 将 ActiveMQ 代理与 LDAP 集成
- 创建和管理 ActiveMQ 代理用户

创建和配置 ActiveMQ 代理

代理 是运行在 Amazon MQ 上的消息代理环境。它是 Amazon MQ 的基本构建块。代理实例类（`m5`、`t3`）和大小（`large`、`micro`）的综合描述是一个代理实例类型（例如 `mq.m5.large`）。有关更多信息，请参阅 代理。

第一个也是最常见的 Amazon MQ 任务是创建代理。以下示例演示如何使用 AWS Management Console创建和配置代理。

主题

步骤 1：配置基本代理设置

1. 登录 Amazon MQ 控制台。
2. 在 Select broker engine (选择代理引擎) 页面上，选择 Apache ActiveMQ。
3. 在 Select deployment and storage (选择部署和存储) 页面的 Deployment mode and storage type (部署模式和存储类型) 部分，执行以下操作：

   a. 选择 Deployment mode (部署模式)（例如 Active/standby broker (主动/备用代理)）。有关更多信息，请参阅 Broker Architecture。

      - 单实例代理由一个可用区中的一个代理组成。代理与您的应用程序以及 Amazon EBS 或 Amazon EFS 存储卷进行通信。有关更多信息，请参阅 Amazon MQ 单实例代理。
      - 高可用性的主动/备用代理由两个不同可用区中的两个代理组成，以冗余对配置。这些代理与您的应用程序以及 Amazon EFS 进行同步通信。有关更多信息，请参阅 用于实现高可用性的 Amazon MQ 主动/备用代理。
      - 有关代理网络示例蓝图的更多信息，请参阅示例蓝图。

   b. 选择 Storage type (存储类型)（例如 EBS）。有关更多信息，请参阅 Storage。

      > **Note**
      >
      > Amazon EBS 在单个可用区内复制数据，但不支持 ActiveMQ 主动/备用部署模式。

c. 选择下一步。

4. 在 Configure settings (配置设置) 页面的 Details (详细信息) 部分，执行以下操作：

a. 输入 Broker name (代理名称)。

> **Important**
>
> 请勿在代理名称中添加个人身份信息（PII）或其他机密或敏感信息。代理名称可供其他 AWS 服务（包括日 CloudWatch 志）访问。代理名称不适合用于私有或敏感数据。

b. 选择 Broker instance type (代理实例类型)（例如 mq.m5.large）。有关更多信息，请参阅 Broker instance types。

5. 在 ActiveMQ Web Console access (ActiveMQ Web 控制台访问) 部分，提供 Username (用户名) 和 Password (密码)。以下限制适用于代理用户名和密码：

- 用户名只能包含字母数字字符、短划线、句点、下划线和波浪线（- . _ ~）。
- 密码必须至少为 12 个字符，包含至少 4 个唯一字符，并且不得包含逗号、冒号或等号（,:=）。

> **Important**
>
> 请勿在代理用户名中添加个人身份信息（PII）或其他机密或敏感信息。其他 AWS 服务（包括 CloudWatch 日志）可以访问经纪人的用户名。代理用户名不适合用于私有或敏感数据。

步骤 2：（可选）配置其他代理设置

> **Important**
>
> - 子网 – 单实例代理需要一个子网（例如默认子网）。一个主动/备用代理需要两个子网。
> - 安全组 – 单实例代理和主动/备用代理都需要至少一个安全组（例如默认安全组）。
> - VPC – 代理的子网和安全组必须在同一个 VPC 中。不支持 EC2-Classic 资源。Amazon MQ 仅支持默认 VPC 租赁，不支持专用 VPC 租赁。
> - Encryption (加密) – 选择客户主密钥来加密您的数据。请参阅 静态加密。

- 公开可用性 – 禁用公开可用性会使代理只能在您的 VPC 中访问。有关更多信息，请参阅首选不可公开访问的代理和访问不可公开访问的代理 Web 控制台。

1. 展开其他设置部分。
2. 在 Configuration (配置) 部分中，选择 Create a new configuration with default values (使用默认值创建新配置) 或 Select an existing configuration (选择现有配置)。有关更多信息，请参阅配置和Amazon MQ Broker Configuration Parameters。
3. 在日志部分，选择是否将常规日志和审核日志发布到 Amazon L CloudWatch ogs。有关更多信息，请参阅 Configuring Amazon MQ to publish logs to Amazon CloudWatch Logs。

> **Important**
>
> 如果您未在 Amazon MQ 用户创建或重启代理之前将 `CreateLogGroup` 权限添加到 Amazon MQ 用户，则 Amazon MQ 不会创建日志组。
> 如果您没有为 Amazon MQ 配置基于资源的策略，则代理无法将日志发布到日志 CloudWatch 。

4. 在 Network and security (网络和安全性) 部分中，配置您的代理的连接：

   a. 请执行以下操作之一：

      - 选择 Use the default VPC, subnet(s), and security group(s) (使用默认 VPC、子网和安全组)。
      - 选择 Select existing VPC, subnet(s), and security group(s) (选择现有 VPC、子网和安全组)。

        1. 如果选择此选项，则可以在 Amazon VPC 控制台上创建一个新的 Virtual Private Cloud (VPC)，并选择现有 VPC 或选择默认 VPC。有关更多信息，请参阅《Amazon VPC 用户指南》中的什么是 Amazon VPC？。
        2. 创建或选择 VPC 后，您可以在 Amazon VPC 控制台上创建新的 Subnet(s) (子网) 或选择现有子网。有关更多信息，请参阅 Amazon VPC 用户指南 中的 VPC 和子网。
        3. 创建或选择子网后，您可以选择 Security group(s) (安全组)。

   b. 选择要用于加密您的数据的客户主密钥 (CMK)。请参阅 静态加密。

   c. 选择您的代理的 Public accessibility (公开可用性)。

5. 在 Maintenance (维护) 部分中，配置您的代理的维护计划：

a. 要在 Apache 发布新版本时将代理升级到新版本，请选择 Enable automatic minor version upgrades (启用自动次要版本升级)。自动升级在维护时段内发生，该维护时段使用星期几、几点（24 小时格式）和时区（默认为 UTC）定义。

> **Note**
>
> 对于主动/备用代理，如果其中一个代理实例正在维护，则 Amazon MQ 会在短时间内使非活动实例停止服务。这允许运行状况良好的备用实例处于活动状态并开始接受传入通信。

b. 请执行以下操作之一：

- 要允许 Amazon MQ 自动选择维护时段，请选择 No preference (无首选项)。
- 要设置自定义维护时段，请选择 Select maintenance window (选择维护时段)，然后指定升级的 Start day (起始日) 和 Start time (起始时间)。

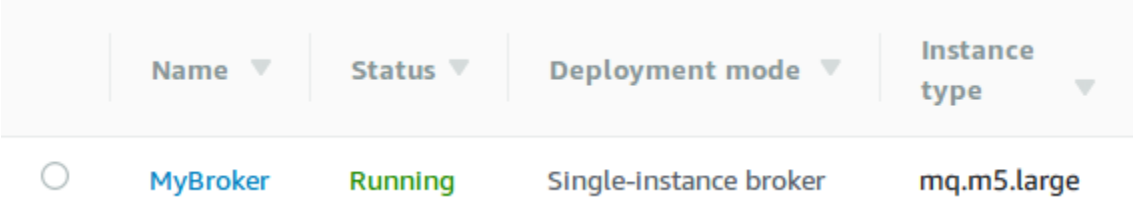
步骤 3：完成创建代理

1. 选择部署。

   当 Amazon MQ 创建您的代理时，会显示 Creation in progress (正在创建) 状态。

   创建代理大约需要 15 分钟。

   成功创建您的代理后，Amazon MQ 会显示 Running (正在运行) 状态。



2. 选择***MyBroker***。

   在该***MyBroker***页面的 Connect 部分，记下您的经纪商的 ActiveMQ 网页控制台网址，例如：

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

   另外，请记下您代理的线级协议终端节点。以下是 OpenWire 终端节点的示例：

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

> **Note**
>
> 对于主动/备用代理，Amazon MQ 提供两个 ActiveMQ Web 控制台 URL，但每次只有一个 URL 处于活动状态。同样，Amazon MQ 为每个线级协议提供两个终端节点，但每次每对中只有一个终端节点处于活动状态。-1 和 -2 后缀表示冗余对。有关更多信息，请参阅 Broker Architecture。
> 对于线级协议终端节点，您可以允许应用程序使用故障转移传输连接到任一终端节点。

编辑代理引擎版本、实例类型、 CloudWatch 日志和维护首选项

除了编辑代理配置和管理配置修订，您还可以配置特定于代理的首选项。

> **Note**
>
> 除自动次要版本升级之外的所有首选项都要求您安排修改。有关更多信息，请参阅 Amazon MQ 代理配置生命周期。

以下示例演示如何使用 AWS Management Console编辑 Amazon MQ ActiveMQ 代理首选项。

编辑 ActiveMQ 代理选项

1. 登录 Amazon MQ 控制台。
2. 从经纪商列表中，选择您的经纪商（ 例如 MyBroker ），然后选择编辑。
3. 在编辑***MyBroker***页面的规格部分，选择代理引擎版本或代理实例类型。
4. 在 Configuration (配置) 部分中，选择代理的配置和修订。有关更多信息，请参阅 Creating and applying broker configurations。
5. 在 Security and network (安全与网络) 部分中，从 Security group(s) (安全组) 下拉列表中选择一个组，或者选择 Create a new security group (创建新的安全组) 以打开 Amazon VPC 控制台。
6. 在CloudWatch 日志部分，选择是否将常规日志和审核日志发布到 Amazon L CloudWatch ogs。

有关为 ActiveMQ 代理配置 CloudWatch 日志的更多信息，请参阅。Configuring Amazon MQ to publish logs to Amazon CloudWatch Logs

> **Important**
>
> 如果您未在 Amazon MQ 用户创建或重启代理之前将 `CreateLogGroup` 权限添加到 Amazon MQ 用户，则 Amazon MQ 不会创建日志组。
> 如果您没有为 Amazon MQ 配置基于资源的策略，则代理无法将日志发布到日志 CloudWatch 。

7. 在 Maintenance (维护) 部分中，配置您的代理的维护计划：

   要在代理发布时将其升级到新 AWS 版本，请选择 “启用自动次要版本升级”。自动升级在维护时段内发生，该维护时段使用星期几、几点（24 小时格式）和时区（默认为 UTC）定义。

   > **Note**
   >
   > 对于主动/备用代理，如果其中一个代理实例正在维护，则 Amazon MQ 会在短时间内使非活动实例停止服务。这允许运行状况良好的备用实例处于活动状态并开始接受传入通信。

8. 选择 Schedule modifications (计划修改)。

   > **Note**
   >
   > 如果您仅选择 Enable automatic minor version upgrades (启用自动次要版本升级)，该按钮将变为 Save (保存)，因为不必重启代理。

   您的首选项将在指定的时间应用到您的代理。

创建和配置 Amazon MQ 代理网络

代理网络由多个同时活动的单实例代理或主动/备用代理组成。您可以根据应用程序的需求（例如高可用性和可扩展性），采用各种拓扑（例如，集中器、中心辐射型、树形或网格）配置代理网络。例如，中心辐射型代理网络可以提高弹性，并在无法访问某个代理时保留消息。具有集中器拓扑的代理网络可以从接受传入消息的大量代理处收集消息，并将消息集中到更中心的代理，以更好地处理许多传入消息负载。在本教程中，您将了解如何使用源和接收器拓扑来创建一个具有两个代理的代理网络。

有关概念概述和详细配置信息，请参阅以下内容：

- Amazon MQ 代理网络
- 正确配置您的代理网络
- networkConnector
- ##ConnectionStart##
- ActiveMQ 文档中的代理网络

您可以使用 Amazon MQ 控制台创建 Amazon MQ 代理网络。因为您可以开始并行创建这两个代理，所以此过程大约需要 15 分钟。

主题

先决条件

要创建代理网络，您必须具备以下条件：

- 两个或多个同时处于活动状态的代理（在本教程中命名为 MyBroker1 和 MyBroker2）。有关创建代理的更多信息，请参阅Creating and configuring a broker。
- 这两个代理必须位于相同的 VPC 或对等的 VPC 中。有关 VPC 的更多信息，请参阅《Amazon VPC 用户指南》中的什么是 Amazon VPC？和《Amazon VPC 对等连接指南》中的什么是 VPC 对等连接？。

> **Important**
>
> 如果您没有默认 VPC、子网或安全组，则必须先创建它们。有关更多信息，请参阅《Amazon VPC 用户指南》中的以下内容：
>
> - 创建默认 VPC
> - 创建默认子网
> - 正在创建安全组

- 两个用户对两个代理程序具有相同的登录凭证。有关创建用户的更多信息，请参见 创建和管理 ActiveMQ 代理用户。

> **Note**
>
> 将 LDAP 身份验证与代理网络集成时，请确保该用户既作为 ActiveMQ 代理，也作为 LDAP 用户存在。

以下示例使用两个单实例代理。但是，您可以使用主动/备用代理或代理部署模式的组合来创建代理网络。

步骤 1：允许代理之间的流量

创建代理后，必须允许它们之间的流量。

1. 在 Amazon MQ 控制台上，在第 MyBroker2 页的 “详细信息” 部分的 “安全和网络” 下，选择您的安全组的名称
   或。

   此时将显示 EC2 Dashboard 的 Security Groups (安全组) 页面。
2. 从安全组列表中，选择您的安全组。
3. 在页面底部，选择 Inbound (入站)，然后选择 Edit (编辑)。
4. 在编辑入站规则对话框中，为 OpenWire 终端节点添加规则。

   a. 选择添加规则。

   b. 对于 Type (类型)，选择 Custom TCP (自定义 TCP)。

   c. 在 “端口范围” 中，键入 OpenWire 端口 (61617)。

   d. 请执行以下操作之一：

      - 如果您想要限制访问特定 IP 地址，对于 Source (源)，请将 Custom (自定义) 选定，然后输入主机的 IP 地址 `MyBroker1`，然后输入 `/32`。（这会将 IP 地址转换为有效的 CIDR 记录）。有关更多信息，请参阅弹性网络接口。

> **Tip**
>
> 要检索 MyBroker1 的 IP 地址，请在 Amazon MQ 控制台上，选择代理的名称并导航到 Details (详细信息) 部分。

- 如果所有代理都是私有的，并且属于相同的 VPC，则对于 Source (源)，请将 Custom (自定义) 选定，然后键入要编辑的安全组的 ID。

> **Note**
>
> 对于公有代理，您必须使用 IP 地址限制访问。

e. 选择保存。

您的代理现在可以接受入站连接。

步骤 2：为您的代理配置网络连接器

在允许代理之间的流量后，必须为其中一个代理配置网络连接器。

1. 编辑代理 MyBroker1 的配置修订。

   a. 在 MyBroker1 页上，选择编辑。

   b. 在 “编辑 MyBroker 1” 页面的 “配置” 部分，选择 “查看”。

   将会显示配置所使用的代理引擎类型和版本（例如，Apache ActiveMQ 5.15.0）。

   c. 在 Configuration details 选项卡上，会显示配置修订号、描述和 XML 格式的代理配置。

   d. 选择 Edit configuration (编辑配置)。

   e. 在配置文件的底部，取消注释 <networkConnectors> 部分并包含以下信息：

   - 网络连接器的 name。
   - 两个代理共有的 ActiveMQ Web 控制台username。
   - 启用 duplex 连接。
   - 请执行以下操作之一：
     - 如果您要将代理连接到单实例代理，请使用static:前缀和 OpenWire 终端节点uri。MyBroker2例如：

```
<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>
```

- 如果您要将代理连接到活动/备用代理，请使用带有以下查询参数的两个代理uri的static+failover传输和 OpenWire终端节点。？randomize=false&maxReconnectAttempts=0例如：

```
<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**
>
> 不要包含 ActiveMQ 用户的登录凭证。

f. 选择保存。

g. 在 Save revision (保存修订) 对话框中，键入 `Add network of brokers connector for MyBroker2`。

h. 选择 Save (保存) 以保存配置的新修订。

2. 编辑 MyBroker1 以将最新的配置修订设置为立即应用。

a. 在 MyBroker1 页上，选择编辑。

b. 在 “编辑 MyBroker 1” 页面的 “配置” 部分，选择 “计划修改”。

c. 在 Schedule broker modifications (计划代理修改) 部分中，选择 Immediately (立即) 应用修改。

d. 选择 应用。

MyBroker1 会重新启动，而且会应用您的配置修订。

将会创建代理网络。

后续步骤

配置代理网络后，可以通过生成和使用消息来测试它。

> **Important**
>
> 确保在端口 8162（用于 ActiveMQ Web 控制台）和端口 61617（用于端点）*MyBroker1*上为代理启用来自本地计算机的入站连接。 OpenWire
> 您可能还需要调整安全组设置，以允许创建者和使用者连接到代理网络。

1. 在 Amazon MQ 控制台上，导航到 Connections (连接) 部分，记下代理 `MyBroker1` 的 ActiveMQ Web 控制台终端节点。
2. 导航到代理 `MyBroker1` 的 ActiveMQ Web 控制台。
3. 要验证网桥是否已连接，请选择 Network (网络)。

   在 Network Bridges（网桥）部分中，`MyBroker2` 的名称和地址列在 Remote Broker（远程代理）和 Remote Address（远程地址）列中。
4. 从任何可以访问代理 `MyBroker2` 的计算机上，创建一个使用者。例如：

   ```
   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
   ```

   使用者连接到的 OpenWire 终端节点`MyBroker2`并开始使用队列中的消息`MyQueue`。
5. 从任何可以访问代理 `MyBroker1` 的计算机上，创建一个创建者并发送一些消息。例如：

   ```
   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
```

生产者连接到的 OpenWire 终端节点，MyBroker1并开始生成要排队的持久消息MyQueue。

将 Java 应用程序连接到您的 Amazon MQ 代理

创建 Amazon MQ ActiveMQ 代理后，您可以将应用程序连接到该代理。以下示例演示如何使用 Java Message Service（JMS）创建代理连接、创建队列以及发送消息。有关完整的可用 Java 示例，请参阅Working Java Example。

您可以使用各种 ActiveMQ 客户端连接到 ActiveMQ 代理。我们建议使用 ActiveMQ 客户端。

主题

- 先决条件
- 创建消息创建者并发送消息
- 创建消息使用者并接收消息

先决条件

启用 VPC 属性

要确保您的代理可以在您的 VPC 中访问，您必须启用 enableDnsHostnames 和 enableDnsSupport VPC 属性。有关更多信息，请参阅《Amazon VPC 用户指南》中的 VPC 中的 DNS Support。

启用入站连接

1. 登录 Amazon MQ 控制台。
2. 从经纪人列表中，选择您的经纪商的名称（例如 MyBroker）。
3. 在该***MyBroker***页面的 “连接” 部分，记下代理的 Web 控制台 URL 和线级协议的地址和端口。
4. 在 Details (详细信息) 部分的 Security and network (安全与网络) 下，选择您的安全组名称或 。

   此时将显示 EC2 Dashboard 的 Security Groups (安全组) 页面。
5. 从安全组列表中，选择您的安全组。

6. 在页面底部，选择 Inbound (入站)，然后选择 Edit (编辑)。
7. 在 Edit inbound rules (编辑入站规则) 对话框中，为希望公开访问的每个 URL 或终端节点添加规则（以下示例显示如何为代理 Web 控制台执行此操作。

   a. 选择添加规则。
   b. 对于 Type (类型)，选择 Custom TCP (自定义 TCP)。
   c. 对于 Port Range (端口范围)，键入 Web 控制台端口（8162）。
   d. 对于 Source (源)，选择 Custom (自定义)，然后键入您希望能够访问 Web 控制台的系统的 IP 地址（例如 `192.0.2.1`)。
   e. 选择保存。

   您的代理现在可以接受入站连接。

添加 Java 依赖项

将 `activemq-client.jar` 和 `activemq-pool.jar` 程序包添加到 Java 类路径中。以下示例说明了 Maven 项目的 `pom.xml` 文件中的这些依赖关系。

```
<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>
```

有关 `activemq-client.jar` 的更多信息，请参阅 Apache ActiveMQ 文档中的初始配置。

> **Important**
>
> 在以下示例代码中，生产者和使用者在单个线程中运行。对于生产系统（或测试代理实例故障转移），请确保您的创建者和使用者在单独的主机或线程上运行。

创建消息创建者并发送消息

1. 使用代理的终端节点为消息创建者创建 JMS 池连接工厂，然后针对该工厂调用 `createConnection` 方法。

> **Note**
>
> 对于主动/备用代理，Amazon MQ 提供两个 ActiveMQ Web 控制台 URL，但每次只有一个 URL 处于活动状态。同样，Amazon MQ 为每个线级协议提供两个终端节点，但每次每对中只有一个终端节点处于活动状态。`-1` 和 `-2` 后缀表示冗余对。有关更多信息，请参阅 Broker Architecture。
> 对于线级协议终端节点，您可以允许应用程序使用故障转移传输连接到任一终端节点。

```
// 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**
>
> 消息创建者应始终使用 `PooledConnectionFactory` 类。有关更多信息，请参阅 始终使用连接池。

2. 创建一个会话，一个名为 MyQueue 的队列和消息创建者。

```
// 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. 创建消息字符串 "Hello from Amazon MQ!"，然后发送消息。

```
// 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. 清理创建者。

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

创建消息使用者并接收消息

1. 使用代理的终端节点为消息创建者创建 JMS 连接工厂，然后针对该工厂调用 createConnection 方法。

```
// 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**
>
> 消息使用者绝不 应使用 PooledConnectionFactory 类。有关更多信息，请参阅 始终使用连接池。

2. 创建一个会话，一个名为 MyQueue 的队列和消息使用者。

```
// 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. 开始等待消息,并在消息到达时收到消息。

```
// 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**
>
> 与 AWS 消息服务（例如 Amazon SQS）不同，消费者经常与经纪人建立联系。

4. 关闭使用者、会话和连接。

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

```
consumerConnection.close();
```

将 ActiveMQ 代理与 LDAP 集成

> **Important**
>
> RabbitMQ 代理不支持 LDAP 集成。

您可以使用以下启用 TLS 的协议来访问 ActiveMQ 代理：

- AMQP
- MQTT
- 基于 WebSocket 的 MQTT
- OpenWire
- STOMP
- 基于 WebSocket 的 STOMP

Amazon MQ 提供以下用户权限管理选项：本机 ActiveMQ 身份验证和 LDAP 身份验证以及授权。有关与 ActiveMQ 用户名和密码相关的限制的信息，请参阅用户。

要授权 ActiveMQ 用户和组使用队列和主题，必须编辑您的代理的配置。Amazon MQ 使用 ActiveMQ 的简单身份验证插件来限制读取和写入到目标。有关详细信息和示例，请参阅始终配置授权映射和 `authorizationEntry`。

> **Note**
>
> 目前，Amazon MQ 不支持客户端证书身份验证。

主题

将 LDAP 与 ActiveMQ 集成

您可以通过存储在轻型目录访问协议（LDAP）服务器中的凭证对 Amazon MQ 用户进行身份验证。您还可以添加、删除和修改 Amazon MQ 用户，并借此为主题和队列分配权限。创建、更新和删除代理等管理操作仍然需要 IAM 凭证，并且未与 LDAP 集成。

希望使用 LDAP 服务器简化和集中化 Amazon MQ 代理身份验证和授权的客户可以使用此功能。将所有用户凭证保留在 LDAP 服务器中，为存储和管理这些凭证提供了一个中心位置，从而节省了时间和精力。

Amazon MQ 使用 Apache ActiveMQ JAAS 插件来提供 LDAP 支持。该插件支持的任何 LDAP 服务器，如 Microsoft Active Directory 或 OpenLDAP，还得到了 Amazon MQ 的支持。有关插件的更多信息，请参阅 Active MQ 文档的 Security (安全) 部分。

除用户外，您还可以通过 LDAP 服务器指定对特定组或用户的主题和队列的访问权限。您可以通过在 LDAP 服务器中创建表示主题和队列的条目，然后为特定 LDAP 用户或组分配权限来实现此目的。然后，您可以将代理配置为从 LDAP 服务器中检索授权数据。

先决条件

在将 LDAP 支持添加到新的或现有的 Amazon MQ 代理之前，您必须设置一个服务账户。启动与 LDAP 服务器的连接需要此服务账户，并且必须具有正确的权限才能建立此连接。此服务账户将为您的代理设置 LDAP 身份验证。任何连续的客户端连接都将通过同一连接进行身份验证。

服务账户是 LDAP 服务器中有权启动连接的账户。这是一个标准的 LDAP 要求，您只能提供一次服务账户凭证。设置连接后，所有未来的客户端连接都将通过 LDAP 服务器进行身份验证。您的服务账户凭证以加密形式安全存储，只有 Amazon MQ 才能访问。

要与 ActiveMQ 集成，LDAP 服务器上需要一个特定的目录信息树（DIT）。有关可清楚显示此结构的 `ldif` 文件示例，请参阅 ActiveMQ 文档 Security (安全) 部分中的将以下 LDIF 文件导入 LDAP 服务器。

LDAP 入门

要开始操作，在创建新的 Amazon MQ 或编辑现有代理实例时，请导航到 Amazon MQ 控制台，并选择 LDAP authentication and authorization (LDAP 身份验证和授权)。

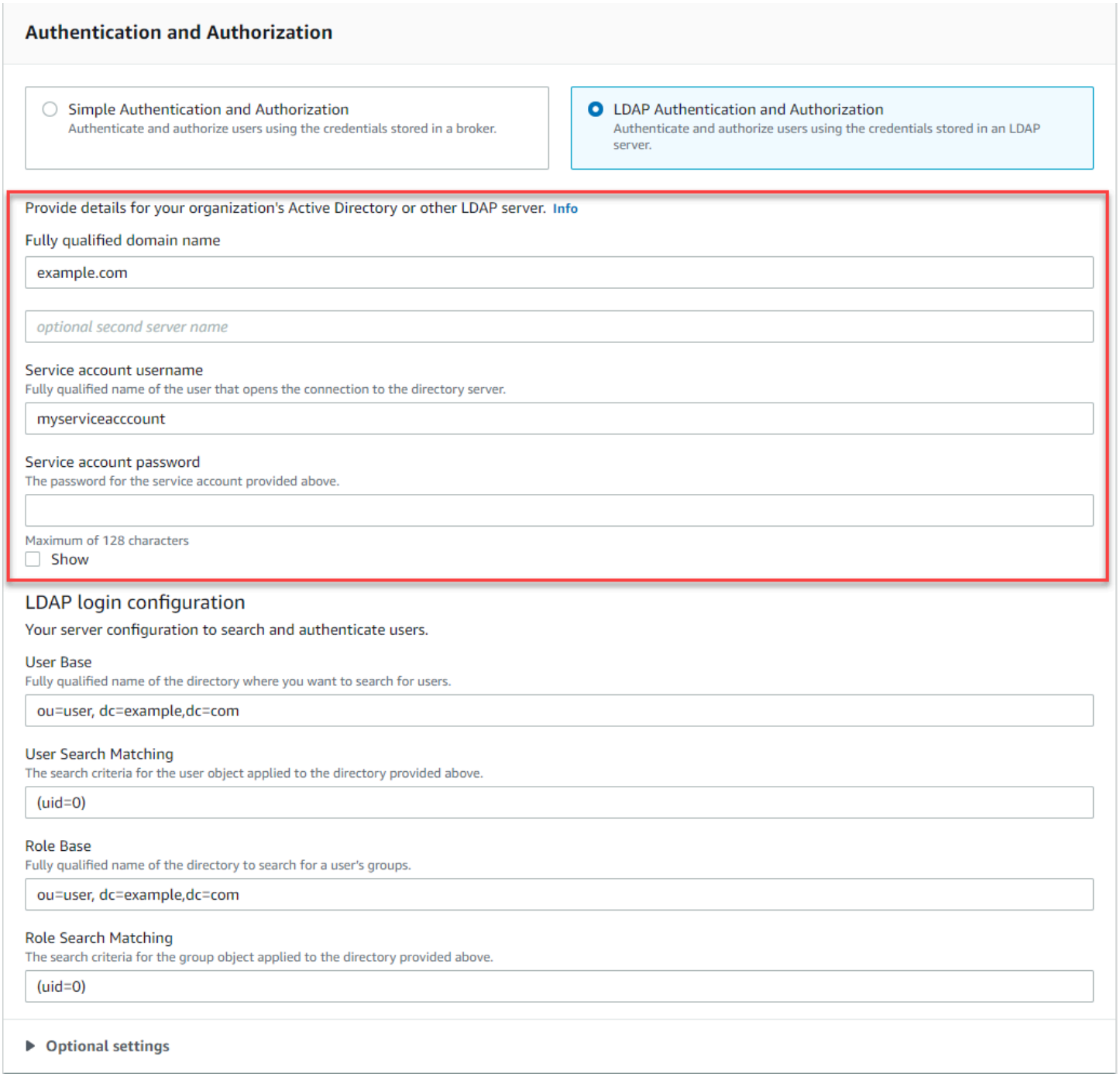
提供有关服务账户的以下信息：

- 完全限定域名：要向其发出身份验证和授权请求的 LDAP 服务器的位置。

> **Note**
>
> 您提供的 LDAP 服务器的完全限定域名不能包含协议或端口号。Amazon MQ 会在完全限定域名前加上协议 `ldaps`，并将附加端口号 636。
> 例如，如果您提供完全限定域 `example.com`，Amazon MQ 将使用以下 URL 访问您的 LDAP 服务器：`ldaps://example.com:636`。
> 为了使代理主机能够与 LDAP 服务器成功通信，完全限定域名必须可以公开解析。要保持 LDAP 服务器的私有和安全性，请在服务器入站规则中限制入站流量，以便仅允许来自代理 VPC 内的流量。

- 服务账户用户名：用于执行与 LDAP 服务器初始绑定的用户的可分辨名称。
- 服务账户密码：执行初始绑定的用户的密码。

下图突出显示了提供这些详细信息的位置。



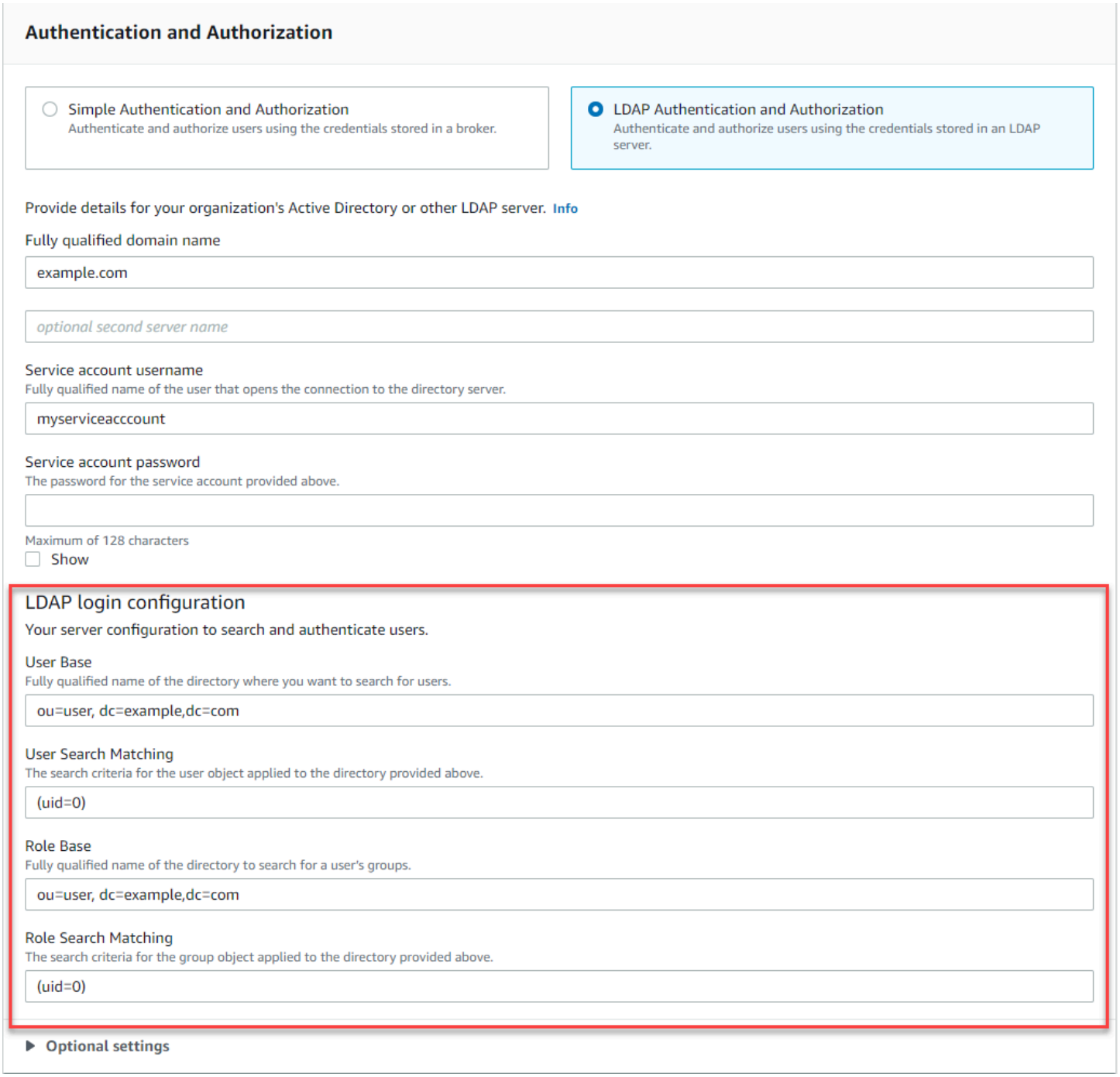
在 LDAP login configuration (LDAP 登录配置) 部分中，提供以下必要信息：

- 用户基础：将为用户搜索的目录信息树（DIT）中的节点的可分辨名称。
- 用户搜索匹配：LDAP 搜索筛选条件，将用于在 `userBase` 中查找用户。客户端的用户名将替换搜索筛选条件中的 `{0}` 占位符。有关更多信息，请参阅身份验证和Authorization。
- 角色基础：将为角色搜索的 DIT 中的节点的可分辨名称。角色可以配置为目录中的显式 LDAP 组条目。典型角色条目可能由角色名称的一个属性（例如公用名（CN））和另一个属性（例如

member）组成，其中的值表示属于角色组的用户的可分辨名称或用户名。例如，鉴于组织部门 group，您可以提供以下可分辨名称：ou=group,dc=example,dc=com。

- 角色搜索匹配：用于在 roleBase 中查找角色的 LDAP 搜索筛选条件。userSearchMatching 匹配的用户的可分辨名称将替换为搜索筛选条件中的 {0} 占位符。客户端的用户名将替换为 {1} 占位符。例如，如果目录中的角色条目包含名为 member 的属性（包含该角色中所有用户的用户名），则可以提供以下搜索筛选条件：(member:=uid={1})。

下图突出显示了指定这些详细信息的位置。



在 Optional settings (可选设置) 部分中，您可以提供以下可选信息：

- 用户角色名称：用户组成员资格的用户目录条目中 LDAP 属性的名称。在某些情况下，用户角色可能由用户目录条目中属性的值来标识。`userRoleName` 选项允许您提供此属性的名称。例如，让我们考虑以下用户条目：

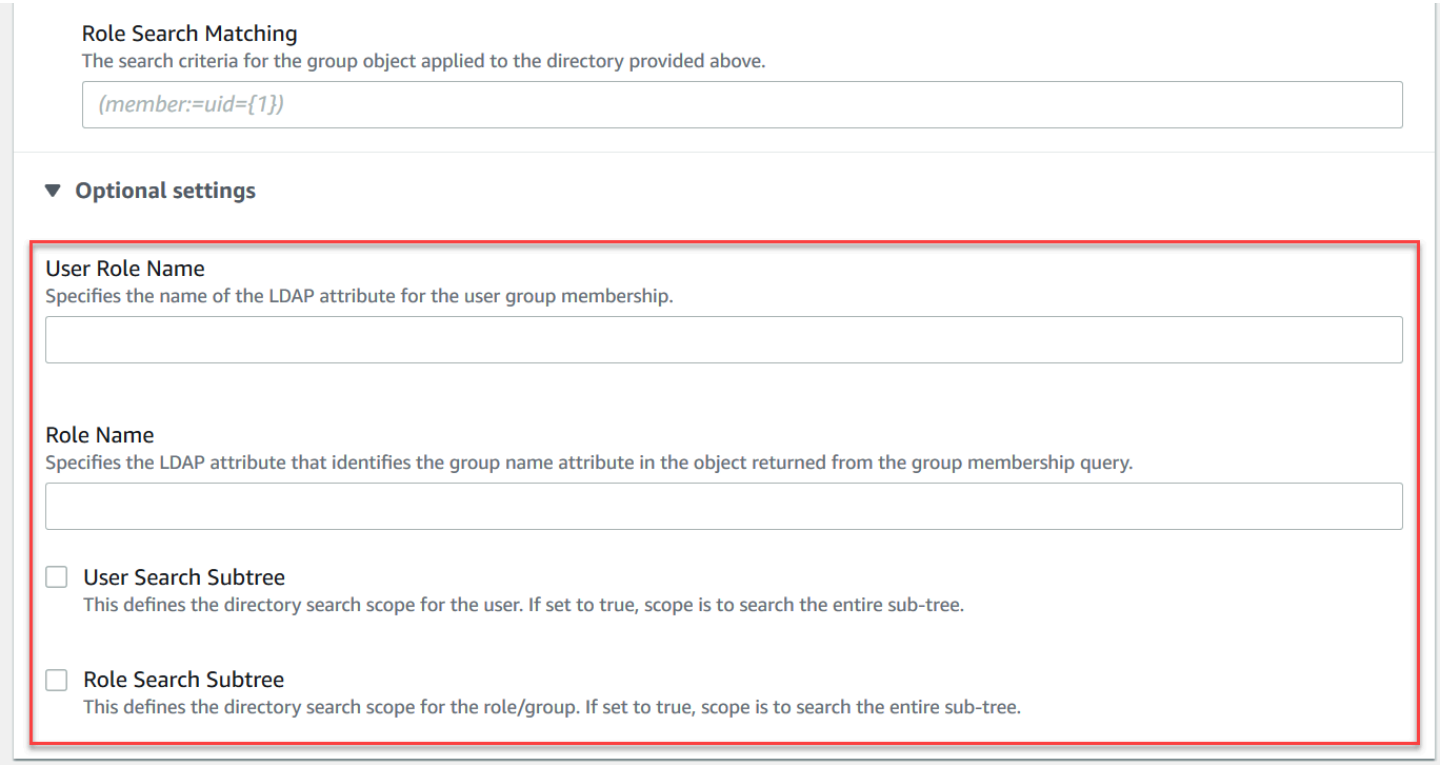
```
dn: uid=jdoe,ou=user,dc=example,dc=com
```

```
objectClass: user
uid: jdoe
sn: jane
cn: Jane Doe
mail: j.doe@somecompany.com
memberOf: role1
userPassword: password
```

要为上述示例提供正确的 `userRoleName`，您需要指定 `memberOf` 属性。如果身份验证成功，则会向用户分配角色 `role1`。

- 角色名称：角色条目中的组名属性，值为该角色的名称。例如，您可以为组条目的通用名称指定 `cn`。如果身份验证成功，则会为用户分配其作为成员的每个角色条目的 `cn` 属性值。
- 用户搜索子树：定义 LDAP 用户搜索查询的范围。如果为 true，则搜索范围设置为由 `userBase` 定义的节点下的整个子树。
- 角色搜索子树：定义 LDAP 角色搜索查询的范围。如果为 true，则搜索范围设置为由 `roleBase` 定义的节点下的整个子树。

下图突出显示了指定这些可选设置的位置。



LDAP 集成的工作方式

您可以将集成分为两大类：身份验证结构和授权结构。

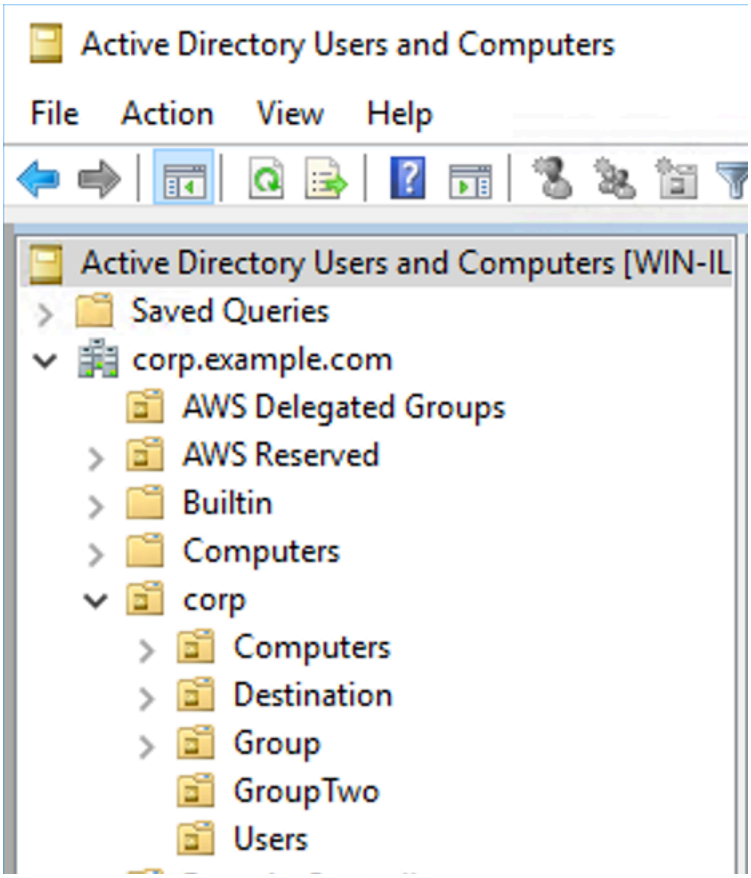
身份验证

对于身份验证，客户端凭证必须有效。这些凭证针对 LDAP 服务器的用户基础中的用户进行验证。

提供给 ActiveMQ 代理的用户基础必须指向 DIT 中存储用户的 LDAP 服务器中的节点。例如，如果您使用的是 AWS Managed Microsoft AD，并且您拥有域组件 `corp`、`example` 和 `com`，而在这些组件中，您拥有组织部门 `corp` 和 `Users`，那么您将使用以下几点作为您的用户基础：

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

ActiveMQ 代理将在 DIT 中的此位置搜索用户，以便对发给代理的客户端连接请求进行身份验证。



由于 ActiveMQ 源代码对用户的属性名称硬编码为 `uid`，您必须确保每个用户都设置了此属性。为简单起见，您可以使用用户的连接用户名。有关更多信息，请参阅 activemq 源代码和在 Windows Server 2016（及后续）版本的 Active Directory 用户和计算机中配置 ID 映射。

要为特定用户启用 ActiveMQ 控制台访问，请确保他们属于 amazonmq-console-admins 组。

Authorization

对于授权，权限搜索基础在代理配置中指定。授权通过代理的 activemq.xml 配置文件中的 cachedLdapAuthorizationMap 元素在每个目标（或通配符，目标集）上完成。有关更多信息，请参阅缓存 LDAP 授权模块。

> **Note**
>
> 为能够使用代理 cachedLDAPAuthorizationMap 配置文件中的 activemq.xml 元素，您必须在通过 AWS Management Console 创建配置时选择 LDAP Authentication and Authorization（LDAP 身份验证和授权）选项，或者在使用 Amazon MQ API 创建新配置时将 authenticationStrategy 属性设置为 LDAP。

您必须提供以下三个属性作为 cachedLDAPAuthorizationMap 元素的一部分：

- queueSearchBase
- topicSearchBase
- tempSearchBase

> **Important**
>
> 为了防止敏感信息直接放置在代理的配置文件中，Amazon MQ 阻止以下属性在 cachedLdapAuthorizationMap 中使用：
>
> - connectionURL
> - connectionUsername
> - connectionPassword
>
> 创建代理时，Amazon MQ 会替换您通过AWS Management Console提供的值，或者在 API 请求的 ldapServerMetadata 属性中，替换上述属性。

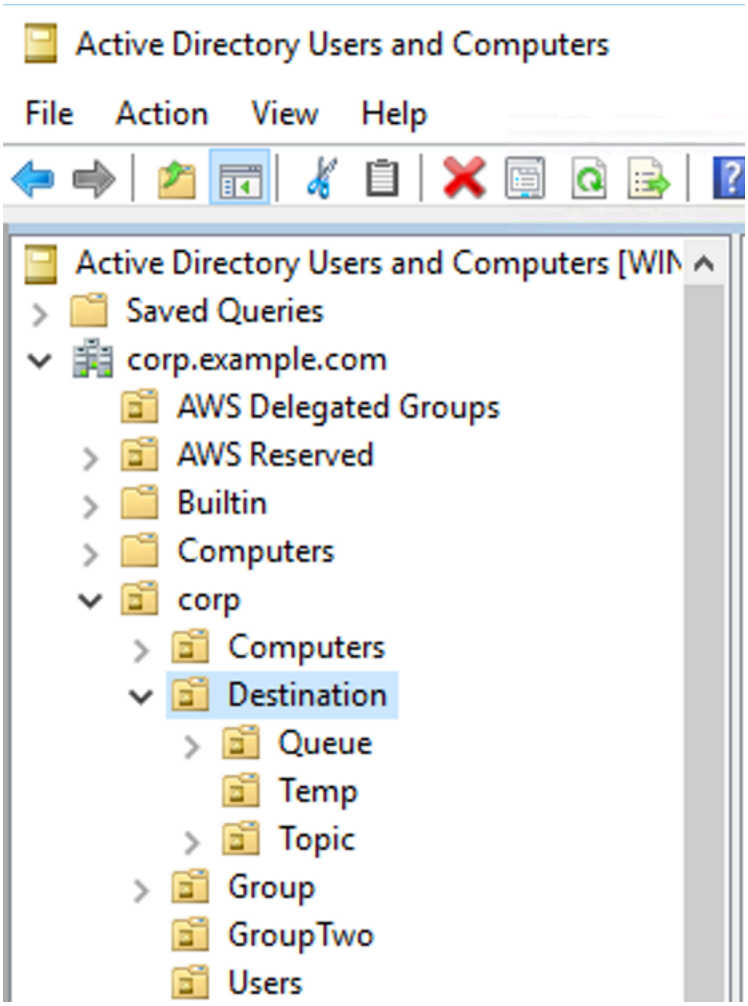
以下演示了 cachedLdapAuthorizationMap 的工作示例。

```
<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>
```

这些值标识 DIT 中为每个目标类型指定权限的位置。因此，对于上面的 AWS Managed Microsoft AD 示例，使用与 `corp`、`example` 和 `com` 相同的域组件，您可以指定一个名为 `destination` 的组织部门，来包含所有目标类型。在该组织部门中，您将为 `queues`、`topics` 和 `temp` 目标分别创建一个。

这意味着您的队列搜索基础（为类型队列的目标提供授权信息）将在 DIT 中具有以下位置：

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



同样，主题和临时目标的权限规则将位于 DIT 中的同一级别：

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

在每个目标类型（队列、主题、临时）的组织部门中，可以提供通配符或特定目标名称。例如，若要为以前缀 DEMO.EVENTS.$. 开头的所有队列提供授权规则，您可以创建以下组织部门：

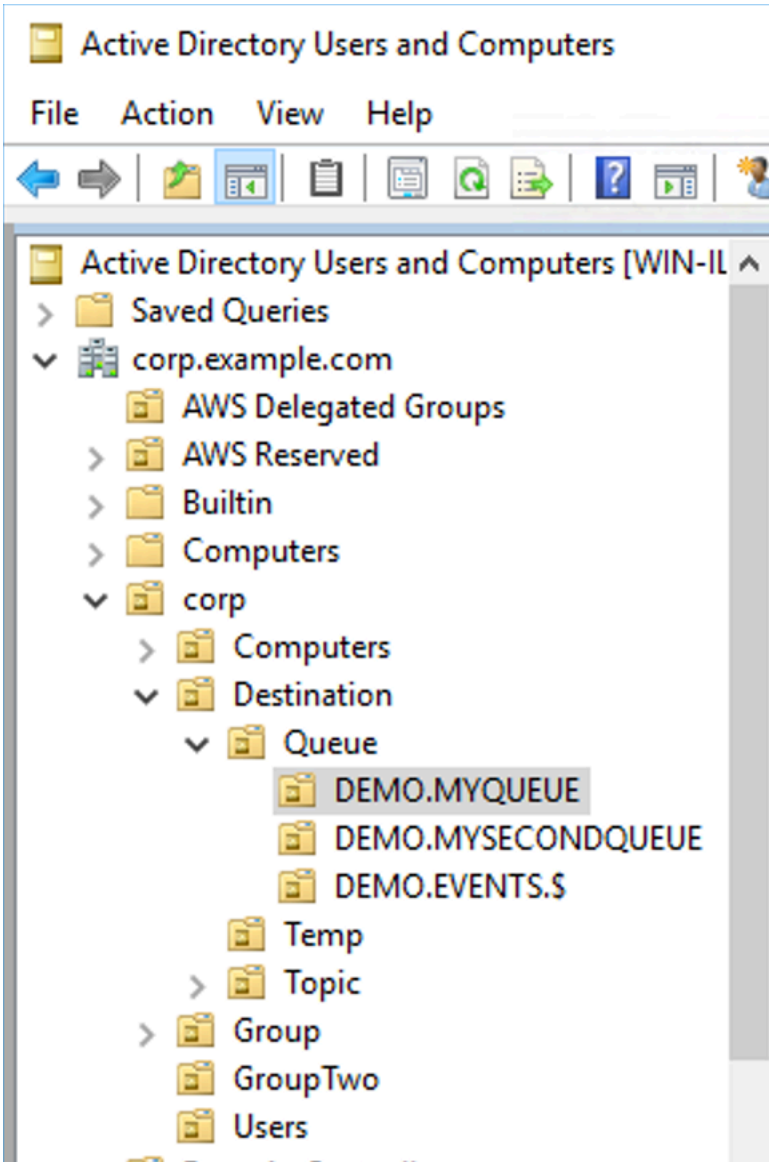
```
OU=DEMO.EVENTS.$,OU=Queue,OU=Destination,OU=corp,DC=corp,DC=example,DC=com
```

> **Note**
>
> DEMO.EVENTS.$ 组织部门位于 Queue 组织部门内。

有关 ActiveMQ 中通配符的更多信息，请参阅 Wildcards (通配符)

要为特定队列提供授权规则（如 DEMO.MYQUEUE），请指定类似以下内容：

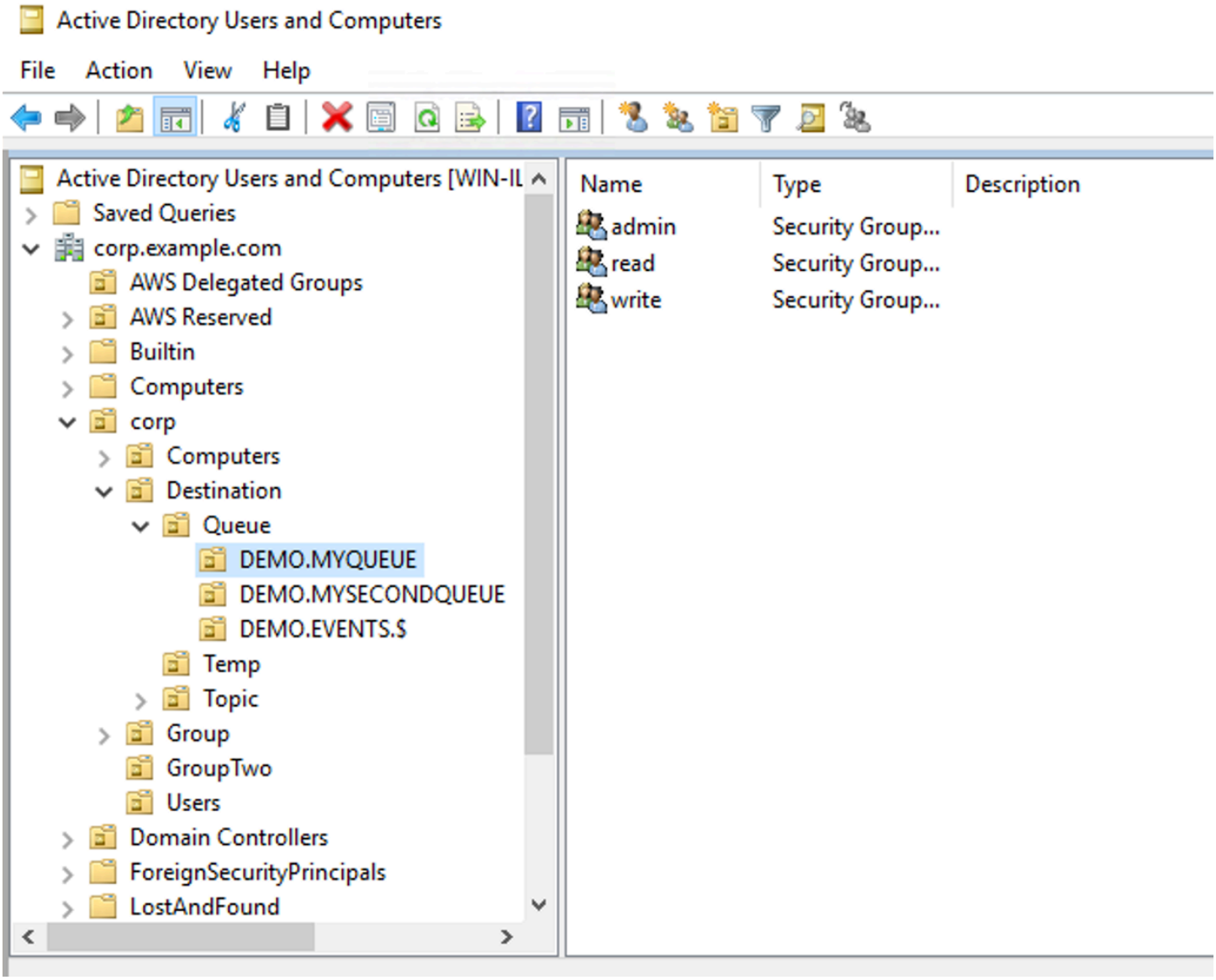
```
OU=DEMO.MYQUEUE,OU=Queue,OU=Destination,OU=corp,DC=corp,DC=example,DC=com
```



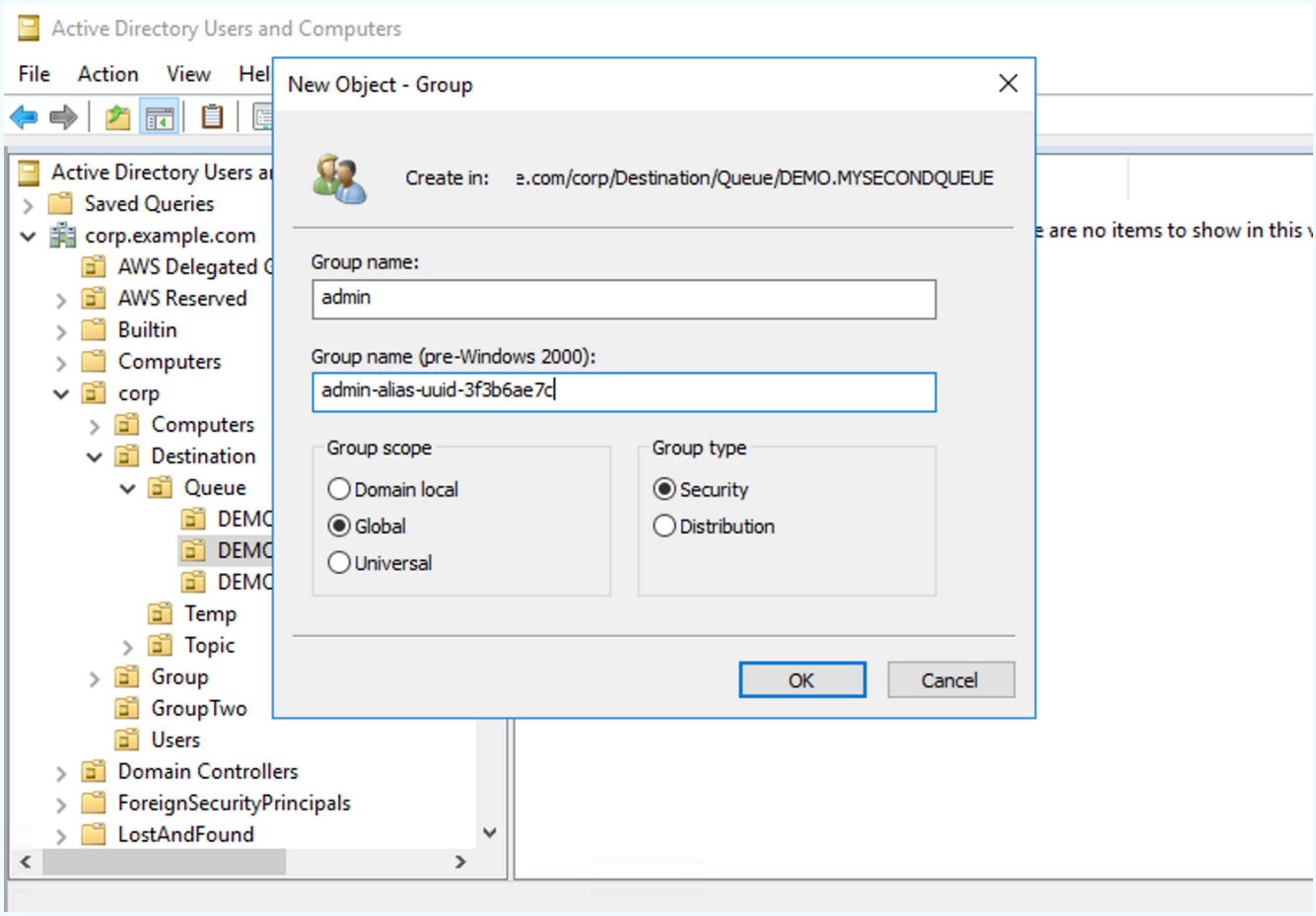
安全组

在每个表示目标或通配符的组织部门中，您必须创建三个安全组。与 ActiveMQ 中的所有权限一样，这些是读/写/管理权限。有关每个权限允许用户执行哪些操作的更多信息，请参阅 ActiveMQ 文档中的安全性。

您必须命名这些安全组 `read`、`write` 和 `admin`。在这些安全组中，您可以添加用户或组，他们将有权执行相关操作。对于每个通配符目标集或单个目标，您都需要这些安全组。



> **Note**
>
> 创建管理组时，将与组名称发生冲突。发生此冲突的原因是，Windows 2000 之前的旧式规则不允许组共享相同的名称，即使这些组位于 DIT 的不同位置。Windows 2000 之前版本文本框中的值对设置没有影响，但必须是全局唯一的。为了避免这一冲突，可以为每个 `admin` 组添加 `uuid` 后缀。



将用户添加到特定目标的 `admin` 安全组将允许用户创建和删除该主题。将他们添加到 `read` 安全组将使他们能够从目标读取，而将他们添加到 `write` 组则将使他们能够写入目标。

除了将单个用户添加到安全组权限之外，您还可以添加整个组。但是，由于 ActiveMQ 再次对组的属性名称进行硬编码，因此必须确保要添加的组具有对象类 `groupOfNames`，如 activemq 源代码中所示。

要执行此操作，请遵循与用户 `uid` 相同的流程。请参阅在 Windows Server 2016（及后续）版本的 Active Directory 用户和计算机中配置 ID 映射。

创建和管理 ActiveMQ 代理用户

ActiveMQ 用户是能够访问 ActiveMQ 代理的队列和主题的人或应用程序。您可以将用户配置为具有特定权限。例如，您可以允许某些用户访问 ActiveMQ Web 控制台。

组是一个语义标签。您可以为用户分配组，并为组配置发送、接收和管理特定队列和主题的权限。

> **Note**
>
> 您无法独立于用户来配置组。在向组标签中添加至少一个用户时，将创建组标签；在从组标签中删除所有用户时，将删除组标签。

以下示例演示如何使用 AWS Management Console创建、编辑和删除 Amazon MQ 代理用户。

主题

创建新用户

1. 登录 Amazon MQ 控制台。
2. 从经纪人列表中，选择您的经纪商名称（例如 MyBroker），然后选择查看详情。

   在该***MyBroker***页面的“用户”部分中，列出了该经纪商的所有用户。

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

3. 选择 创建用户。
4. 在 Create user (创建用户) 对话框中，键入 Username (用户名) 和 Password (密码)。
5. (可选) 键入用户所属的组的名称，用逗号分隔 (例如：`Devs, Admins`)。
6. （可选）要使用户能够访问 ActiveMQ Web 控制台，请选择 ActiveMQ Web Console (ActiveMQ Web 控制台)。
7. 选择 创建用户。

> **Important**
>
> 对用户进行更改不会立即将更改应用于用户。要应用更改，必须等待下一维护时段或者重启代理。有关更多信息，请参阅 Amazon MQ 代理配置生命周期。

编辑现有用户

1. 登录 Amazon MQ 控制台。
2. 从经纪人列表中，选择您的经纪商名称（例如 MyBroker），然后选择查看详情。

   在该***MyBroker***页面的“用户”部分中，列出了该经纪商的所有用户。

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

3. 选择您的登录凭证，然后选择编辑。

   将会显示 Edit user (编辑用户) 对话框。
4. （可选）键入新 Password (密码)。
5. (可选) 添加或删除用户所属的组的名称，用逗号分隔 (例如：`Managers, Admins`)。
6. （可选）要使用户能够访问 ActiveMQ Web 控制台，请选择 ActiveMQ Web Console (ActiveMQ Web 控制台)。
7. 要保存对用户所做的更改，请选择 Done (完成)。

> **Important**
>
> 对用户进行更改不会立即将更改应用于用户。要应用更改，必须等待下一维护时段或者重启代理。有关更多信息，请参阅 Amazon MQ 代理配置生命周期。

删除现有用户

1. 登录 Amazon MQ 控制台。
2. 从经纪人列表中，选择您的经纪商名称（例如 MyBroker），然后选择查看详情。

   在该***MyBroker***页面的“用户”部分中，列出了该经纪商的所有用户。

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

3. 选择您的登录凭据（例如 ***MyUser***），然后选择 “删除”。
4. 要确认删除用户，请在 “删除 ***MyUser***？” 对话框中，选择 “删除”。

> **Important**
>
> 对用户进行更改不会立即将更改应用于用户。要应用更改，必须等待下一维护时段或者重启代理。有关更多信息，请参阅 Amazon MQ 代理配置生命周期。

Amazon MQ for ActiveMQ 最佳实践

以此作为参考快速找到在 Amazon MQ 上使用 ActiveMQ 代理时最大程度提高性能和降低吞吐量成本的建议。

主题

连接到 Amazon MQ

以下设计模式可以提高您的应用程序连接到 Amazon MQ 代理的有效性。

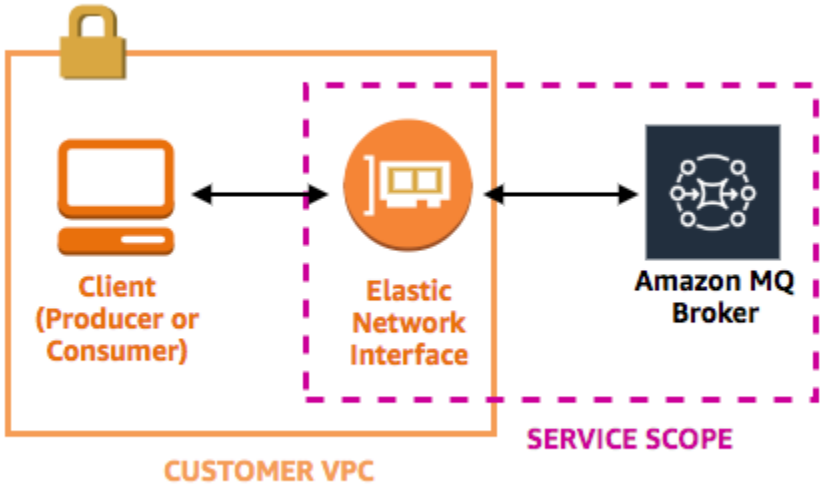
主题

永远不要修改或删除 Amazon MQ 弹性网络接口

在您首次创建 Amazon MQ 代理时，Amazon MQ 会在您账户下的 Virtual Private Cloud（VPC）中预置弹性网络接口，因此需要大量 EC2 权限。该网络接口允许您的客户端（创建者或使用者）与

Amazon MQ 代理通信。该网络接口被视为在 Amazon MQ 的服务范围内，尽管是您的账户的 VPC 的一部分。

> **Warning**
>
> 您不得修改或删除此网络接口。修改或删除网络接口可能会导致永久丢失您的 VPC 和代理之间的连接。



始终使用连接池

在使用单个创建者和单个使用者的方案（例如 Getting Started with Amazon MQ 教程），您可以将单个 `ActiveMQConnectionFactory` 类用于每个创建者和使用者。例如：

```
// 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();
```

但是，在具有多个创建者和使用者的更真实的方案中，为多个创建者创建大量连接可能成本高昂，并且效率低下。在这些方案中，您应使用 PooledConnectionFactory 类将多个创建者请求分组。例如：

> **Note**
>
> 消息使用者绝不 应使用 PooledConnectionFactory 类。

```
// 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();
```

始终使用故障转移传输连接到多个代理终端节点

如果应用程序需要连接到多个代理终端节点，例如，当您使用主/备用部署模式或者从本地消息代理迁移到 Amazon MQ 时，使用故障转移传输以允许您的使用者随机连接到一个。例如：

```
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
```

避免使用消息选择器

可以使用 JMS 选择器将筛选器附加到主题订阅（以将消息基于其内容路由到使用者）。但是，JMS 选择器的使用将会填满 Amazon MQ 代理的筛选器缓冲区，从而阻止其筛选消息。

一般来说，应避免让使用者路由消息，这样做的原因是，为了实现使用者和创建者的最佳解耦，使用者和创建者均应是短暂存在的。

首选虚拟目标而非持久订阅

持久订阅可帮助确保使用者收到发布到主题的所有消息，例如，在恢复丢失的连接后。但是，使用持久订阅还阻止竞争性使用者使用并可能具有大规模性能问题。考虑改用虚拟目标。

如果使用 Amazon VPC 对等连接，请避免 CIDR 范围 **10.0.0.0/16** 内的客户端 IP

如果您要在本地部署基础设施和 Amazon MQ 代理之间设置 Amazon VPC 对等连接，则不得使用 CIDR 范围 `10.0.0.0/16` 内的 IP 配置客户端连接。

确保有效的 Amazon MQ 性能

以下设计模式可以提高 Amazon MQ 代理的效率和性能。

主题

- 对具有慢速使用者的队列禁用并发存储和分派
- 选择正确的代理实例类型以实现最佳吞吐量
- 选择正确的代理存储类型以实现最佳吞吐量
- 正确配置您的代理网络

对具有慢速使用者的队列禁用并发存储和分派

默认情况下，Amazon MQ 针对具有快速使用者的队列进行优化：

- 当使用者能够跟上创建器生成的消息速率时，将其视为快速。
- 如果队列造成了未确认消息积压，并可能导致创建器吞吐量下降，则将使用者视为慢速。

要指示 Amazon MQ 针对具有慢速使用者的队列进行优化，请将 `concurrentStoreAndDispatchQueues` 属性设置为 `false`。有关示例配置，请参阅 `concurrentStoreAndDispatchQueues`。

选择正确的代理实例类型以实现最佳吞吐量

代理实例类型的消息吞吐量取决于应用程序的使用案例和以下因素：

- 持久模式下 ActiveMQ 的使用
- 消息大小
- 创建器和使用者的数量
- 目标的数量

了解消息大小、延迟和吞吐量之间的关系

根据您的使用案例，较大的代理实例类型不一定能提高系统吞吐量。当 ActiveMQ 将消息写入持久存储中，消息的大小决定了系统的限制因素：

- 如果您的消息大小不到 100 KB，则持久性存储延迟是限制因素。
- 如果您的消息大小超过 100 KB，则持久性存储吞吐量是限制因素。

当您在持久模式下使用 ActiveMQ 时，通常会在使用者较少或使用者较慢的情况下发生写入存储。在非持久模式下，如果代理实例的堆内存已满，则也会在使用者较慢的情况下发生写入存储。

要为您的应用程序确定最佳代理实例类型，我们建议您测试不同的代理实例类型。有关详细信息，请参阅Broker instance types以及 Measuring the Throughput for Amazon MQ using the JMS Benchmark。

较大代理实例类型的使用案例

当较大代理实例类型提高吞吐量时，存在以下三个常见使用案例：

- 非持久模式 – 当您的应用程序在代理实例故障转移（例如，播报体育赛事比分时）期间对消息丢失不太敏感时，您通常可以使用 ActiveMQ 的非持久模式。在此模式下，仅在代理实例的堆内存已满时，ActiveMQ 才会将消息写入持久性存储中。使用非持久模式的系统可以从较大代理实例类型所提供的较高内存、较快 CPU 以及较快网速中受益。
- 快速使用者 – 当存在活动使用者且 `concurrentStoreAndDispatchQueues` 标志启用时，ActiveMQ 允许消息直接从创建器传递到使用者，而无需将消息发送到存储（即使在持久模式下也是如此）。如果您的应用程序可以快速使用消息（或者您可以将使用者设计为这么做），则应用程序能从较大代理实例类型中受益。要让您的应用程序更快地使用消息，请为应用程序实例添加使用者线程，或者纵向或横向扩展应用程序实例。
- 批处理事务 – 当您使用持久模式且在每个事务中发送多条消息时，您可以使用较大代理实例类型来实现总体更高消息吞吐量。有关更多信息，请参阅 ActiveMQ 文档中的我是否应使用事务？。

选择正确的代理存储类型以实现最佳吞吐量

要利用跨多个可用区的高持久性和复制功能，请使用 Amazon EFS。要利用低延迟和高吞吐量，请使用 Amazon EBS。有关更多信息，请参阅Storage。

正确配置您的代理网络

当您创建代理网络时，请为您的应用程序正确配置它：

- 启用持续模式 – 因为（相对于其对等项）每个代理实例充当创建者或使用者，所以代理网络不提供消息的分布式复制。第一个充当使用者的代理接收消息并将其保留到存储中。此代理向创建者发送确认，并将消息转发给下一个代理。当第二个代理确认消息的持久性后，第一个代理将删除该消息。

  如果禁用持久模式，则第一个代理会在不将消息保留到存储的情况下确认创建者。有关更多信息，请参阅 Apache ActiveMQ 文档中的复制消息存储和持久和非持久交付有什么区别？。
- 请勿对代理实例禁用建议消息 – 有关更多信息，请参阅 Apache ActiveMQ 文档中的建议消息。
- 请勿使用多播代理发现 – Amazon MQ 不支持使用多播的代理发现。有关更多信息，请参阅 Apache ActiveMQ 文档中的发现、多播和 zeroconf 的区别是什么？。

通过恢复已准备 XA 事务避免缓慢重

ActiveMQ 支持分布式 (XA) 事务。了解 ActiveMQ 如何处理 XA 事务有助于避免 Amazon MQ 中代理重启和故障转移的缓慢恢复时间

每次重启时都会重放未解析的已准备 XA 事务。如果这些问题仍未被解析，其数量将随着时间的推移而增长，从而显著增加启动代理所需的时间。这会影响重启和故障转移时间。您必须使用 `commit()` 或 `rollback()` 解析这些事务，以便性能不会随着时间的推移而降低。

要监控未解析的已准备 XA 事务，您可以使用 Amazon CloudWatch Logs 中的 `JournalFilesForFastRecovery` 指标。如果该数字不断增加，或者始终高于 1，则应使用类似于以下示例的代码恢复未解析的事务。有关更多信息，请参阅Amazon MQ 中的配额。

以下示例代码遍历已准备 XA 事务，并使用 `rollback()` 关闭它们。

```
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) {
        }
    }
}
```

在实际场景中，您可以针对 XA 事务管理器检查已准备 XA 事务。然后，您可以使用 `rollback()` 或 `commit()` 来决定是否处理每个已准备事务。

Amazon MQ for ActiveMQ 的跨区域数据复制

Amazon MQ for ActiveMQ 提供跨区域数据复制（CRDR）功能，此功能允许将消息从主 AWS 区域中的主代理异步复制到副本区域中的副本代理。通过向 Amazon MQ API 发出失效转移请求，当前副本代理提升为主代理角色，而当前主代理降级为副本代理角色。

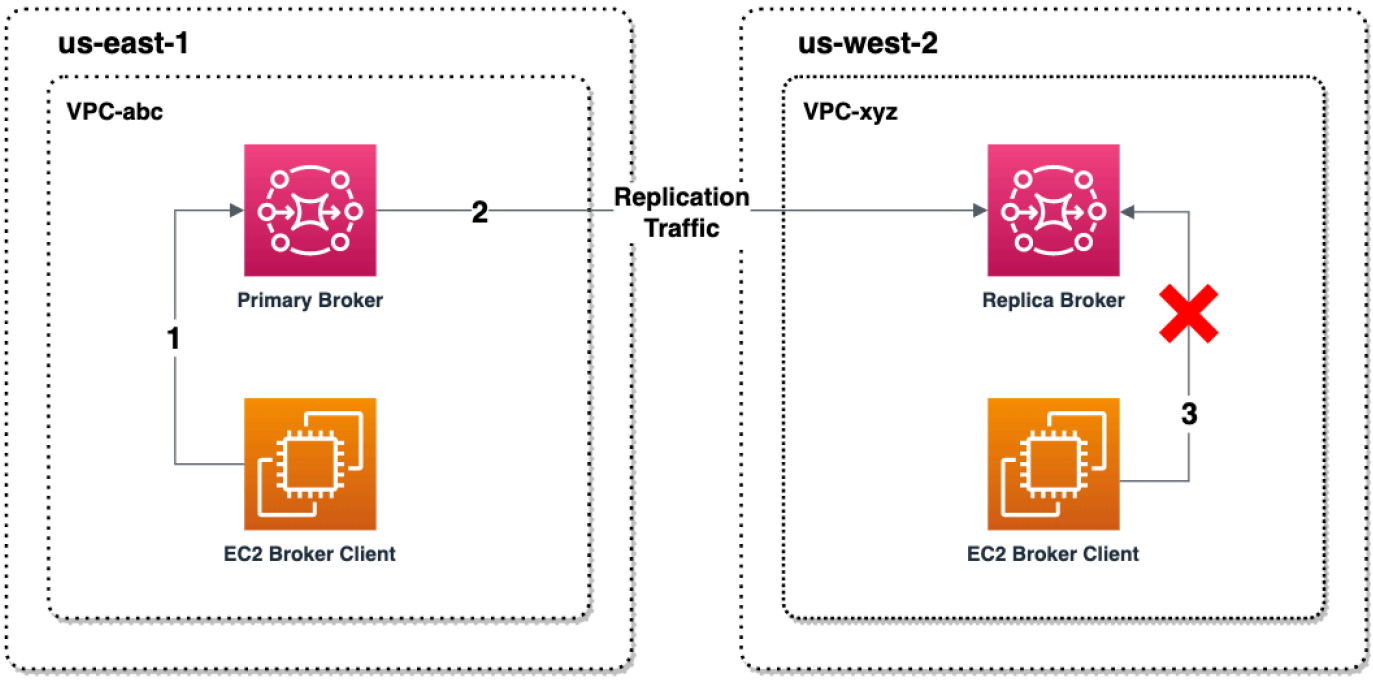
本节提供有关如何使用 Amazon MQ for ActiveMQ 设置跨区域数据复制的教程。

主题

Amazon MQ 中的主代理和副本代理

您可以创建主代理和副本代理，以便将数据从主 AWS 区域中的主代理异步复制到副本区域中的副本代理。主区域由一对冗余的主动/备用代理（称为主代理）组成。辅助区域由一对冗余的主动/备用代理（称为副本代理）组成。

下图说明了辅助区域中的副本代理从主区域中的主代理接收异步复制的数据的过程。



主代理和副本代理充当跨区域数据恢复解决方案。如果主区域中的主代理出现故障，则可以通过启动切换或失效转移，将辅助区域中的副本代理提升为主代理。然后，以前的主代理成为副本代理，而以前的副本代理提升为主代理。有关创建主代理和副本代理的说明，请参阅创建和删除跨区域数据复制代理。

> **Note**
>
> 仅适用于主动/备用代理。

创建和删除跨区域数据复制代理

使用跨区域数据复制（CRDR），您可以根据需要，在两个 AWS 区域中的 Amazon MQ for ActiveMQ 消息代理之间切换。您可以将现有代理指定为主代理并为该代理创建副本，也可以同时创建新的主代理和副本代理。然后，您可以使用 Amazon MQ `Promote` API 操作将副本代理提升为主代理角色。有关主代理和副本代理的更多信息，请参阅Amazon MQ 中的主代理和副本代理。

以下说明描述了如何使用 Amazon MQ 管理控制台创建和配置副本代理。

主题

先决条件

要使用跨区域数据复制功能，您必须查看并遵守以下先决条件：

- 版本：跨区域数据复制功能仅适用于Amazon MQ for ActiveMQ 代理 5.17.6 及更高版本。
- 区域：以下区域支持跨区域数据复制：美国东部（俄亥俄州）、美国东部（弗吉尼亚州北部）、美国西部（俄勒冈州）、美国西部（北加利福尼亚）。
- 实例类型：跨区域数据复制仅适用于代理实例大小 `mq.m5.large` 及更大规模。
- 部署类型：跨区域数据复制仅适用于部署了多可用区的活动/备用代理。
- 代理状态：您只能为代理状态为 `Running` 的主代理创建副本代理。

步骤 1（可选）：创建新的主代理

创建新的主代理

1. 登录 Amazon MQ 控制台。

2. 在 Amazon MQ 控制台的“代理”页面上，选择创建代理。
3. 在 Select broker engine (选择代理引擎) 页面上，选择 Apache ActiveMQ。
4. 在 Select deployment and storage (选择部署和存储) 页面的 Deployment mode and storage type (部署模式和存储类型) 部分，执行以下操作：

   - 对于部署模式，选择主动/备用代理。主动/备用代理由两个不同可用区中配置为冗余对的两个代理组成。这些代理与您的应用程序以及 Amazon EFS 进行同步通信。有关更多信息，请参阅Broker Architecture。
5. 选择下一步。
6. 在 Configure settings (配置设置) 页面的 Details (详细信息) 部分，执行以下操作：

   a. 输入 Broker name (代理名称)。

   > **Important**
   >
   > 请勿在代理名称中添加个人身份信息（PII）或其他机密或敏感信息。其他AWS服务（包括 CloudWatch Logs）可以访问代理名称。代理名称不适合用于私有或敏感数据。

   b. 选择 Broker instance type (代理实例类型)（例如 mq.m5.large）。有关更多信息，请参阅Broker instance types。
7. 在 ActiveMQ Web Console access (ActiveMQ Web 控制台访问) 部分，提供 Username (用户名) 和 Password (密码)。以下限制适用于代理用户名和密码：

   - 用户名只能包含字母数字字符、短划线、句点、下划线和波浪线（- . _ ~）。
   - 密码必须至少为 12 个字符，包含至少 4 个唯一字符，并且不得包含逗号、冒号或等号（,:=）。

   > **Important**
   >
   > 请勿在代理用户名中添加个人身份信息（PII）或其他机密或敏感信息。其他AWS服务（包括 CloudWatch Logs）可以访问代理用户名。代理用户名不适合用于私有或敏感数据。

页面顶部的绿色闪存栏确认 Amazon MQ 正在恢复区域中创建副本代理。您还可以查看代理的 CRDR 角色和 RPO 状态。要关闭“CRDR 角色”和“RPO 状态”列，请选择代理表右上角的齿轮图标。然后，在首选项页面上，关闭 CRDR 角色或 RPO 状态。

步骤 2：创建现有代理的副本

1. 在 Amazon MQ 控制台的“代理”页面上，选择创建副本代理。
2. 在选择主代理页面上，选择要用作 CRDR 主代理的现有代理。然后选择下一步。
3. 在配置副本代理页面上，使用下拉菜单选择副本区域。
4. 在副本代理的 ActiveMQ 控制台用户部分中，提供副本代理控制台用户的用户名和密码。以下限制适用于代理用户名和密码：

   - 用户名只能包含字母数字字符、短划线、句点、下划线和波浪线（ - . _ ~ ）。
   - 密码必须至少为 12 个字符，包含至少 4 个唯一字符，并且不得包含逗号、冒号或等号（ ,:= ）。

   > **Important**
   >
   > 请勿在代理用户名中添加个人身份信息（PII）或其他机密或敏感信息。其他AWS服务（包括 CloudWatch Logs）可以访问代理用户名。代理用户名不适合用于私有或敏感数据。

5. 在用于在代理之间桥接访问的数据复制用户部分中，提供将访问主代理和副本代理的用户的用户名和密码。以下限制适用于代理用户名和密码：

   - 用户名只能包含字母数字字符、短划线、句点、下划线和波浪线（ - . _ ~ ）。
   - 密码必须至少为 12 个字符，包含至少 4 个唯一字符，并且不得包含逗号、冒号或等号（ ,:= ）。

   > **Important**
   >
   > 请勿在代理用户名中添加个人身份信息（PII）或其他机密或敏感信息。其他AWS服务（包括 CloudWatch Logs）可以访问代理用户名。代理用户名不适合用于私有或敏感数据。

   配置任何其他设置。然后选择下一步。
6. 在查看并创建页面上，查看副本代理的详细信息。然后，选择创建副本代理。
7. 接下来，重启主代理。这也将重启副本代理。有关重启代理的说明，请参阅Rebooting a Broker。

有关为 ActiveMQ 代理配置其他设置的更多信息，请参阅创建并连接到 ActiveMQ 代理

删除 CRDR 代理

要删除主或副本 CRDR 代理，必须先取消配对，然后重启代理。以下说明介绍如何使用 AWS 管理控制台取消配对和重启代理。

1. 在代理页面上，选择要取消配对的 CRDR 代理，然后选择编辑。
2. 在数据复制部分的代理编辑页面上，选择取消配对代理。
3. 在弹出窗口中输入“unpair”以确认您的选择。然后选择取消配对代理。
4. 接下来，重启取消配对的主代理。这也将重启副本代理。有关重启代理的说明，请参阅Rebooting a Broker。主代理重启后，两个代理均为取消配对状态，可以单独删除。要删除您的代理，请参阅Deleting a broker。

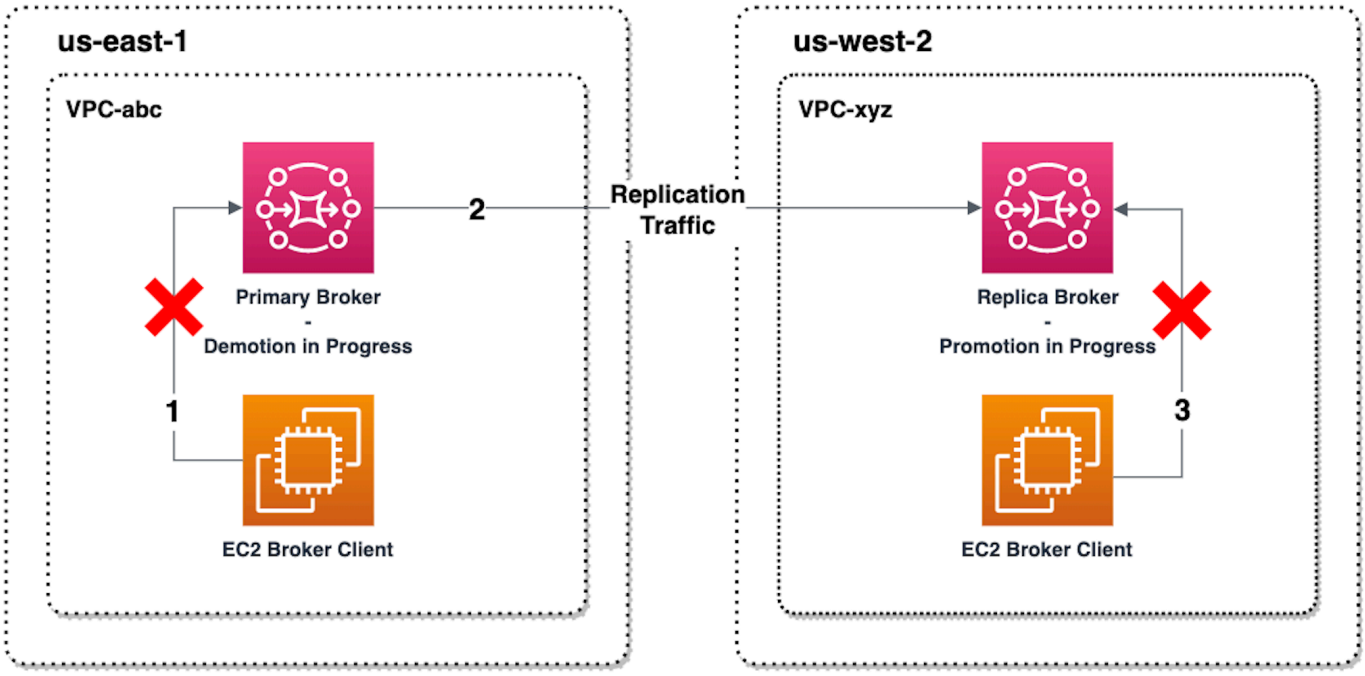
启动切换或失效转移以将副本代理提升为主代理角色

当您想要将副本代理提升为主代理角色时，可以启动切换或失效转移。升级副本代理时，主代理会降级为副本代理角色。

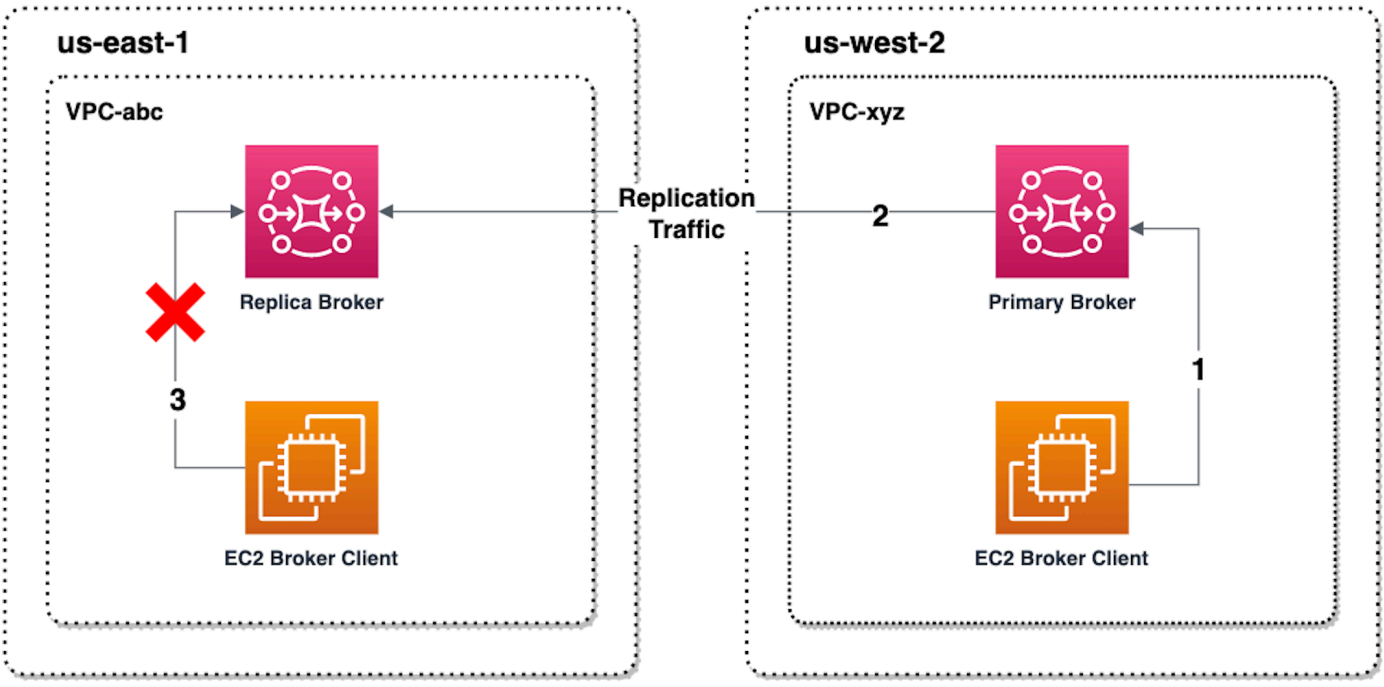
切换优先考虑一致性而不是可用性。当此失效转移操作完成时，可以保证代理处于相同的状态。通过切换，在建立代理间一致性的同时，可能会出现两个代理都无法进行客户端连接的时期。在提升副本的瞬间，这两个代理将处于相同的状态。切换是否成功取决于两个区域的运行状况和区域间网络。

失效转移优先考虑可用性而不是一致性。此操作完成后，不保证经纪商的状态相同。通过失效转移，可以保证副本代理可以立即为客户端流量提供服务，无需等待任何复制数据进行同步或主代理收到关闭信号。失效转移的成功既不取决于原始主区域的运行状况，也不取决于区域间网络。

下图说明了一种切换，在这种切换中，当复制队列耗尽且代理状态同步时，两个代理都不接受客户端连接。在此过程中，主代理 VPC 中的客户端无法在操作进行期间产生进一步的状态更改，并且主代理被降级为副本。当复制队列耗尽且两个代理达到相同状态时，副本代理的 VPC 中的客户端将无法连接到副本代理，直至失效转移操作完成，并且副本代理提升为主代理。



下图说明了切换过程完成后的代理状态。原始副本代理现已提升为主代理角色并正在接受客户端连接。客户可以生成和使用来自代理的数据。



使用控制台提升副本代理

要使用切换或失效转移来提升副本代理，请在 Amazon MQ 控制台中执行以下步骤。

> **Note**
>
> 您无法在主代理上启动切换或失效转移。

1. 切换到副本代理所在区域。在“代理”表中，选择要提升为主代理的现有副本代理。
2. 在代理详细信息页面上，执行以下操作：
    1. 选择提升副本。
    2. 在弹出窗口中，选择切换或失效转移。
    3. 在文本框中输入“确认”以确认您的选择。
    4. 选择确认。

启动失效转移后，代理状态更改为正在进行失效转移。失效转移完成后，“代理”页面顶部的蓝色进度条变为绿色。

> **Note**
>
> 只在创建副本代理时复制配置。不会复制之后的任何更新。

Amazon CloudWatch 中的跨区域数据复制指标

Amazon MQ for ActiveMQ 跨区域数据复制功能提供了用于维护主代理和副本代理的可靠性、可用性和性能的指标。在复制过程中，辅助区域中的副本代理从主区域中的主代理接收异步复制的数据。如果主区域中的主代理出现故障，则可以通过启动切换或失效转移，将辅助区域中的副本代理提升为主代理。有关在 Amazon CloudWatch 中查看指标的说明，请参阅访问 Amazon MQ 的 CloudWatch 指标。

CRDR 时间戳

以下时间戳描述如何计算在 Amazon CloudWatch 中找到的指标。数据复制过程中有五个时间戳：

- 当前观测时间（TCO）：当前瞬间。
- 创建时间（TC）：主代理在复制队列上创建事件的瞬间。在主代理和副本代理上均可用。

- 交付时间（TD)：事件成功交付给副本代理的瞬间。仅在副本代理上可用。
- 处理时间 (TP)：副本代理成功处理事件的瞬间。仅在副本代理上可用。
- 确认时间（TA)：主代理成功确认事件的瞬间。仅在主代理上可用。

使用 CRDR CloudWatch 指标估算切换/失效转移性能

默认情况下，Amazon MQ 将会为您的代理启用指标。您可以通过访问 Amazon CloudWatch 控制台或者通过使用 CloudWatch API 查看您的代理指标。以下指标对于了解 CRDR 代理的复制和切换/失效转移性能很有用：

Amazon MQ CloudWatch 指标	使用 CRDR 的原因
`TotalReplicationLag`	主代理上最后一个未确认事件的 TA 和 TC 之间的估计时间。
`ReplicationLag`	副本代理上最后一个未确认事件的 TP 和 TC 之间的估计时间。
`PrimaryWaitTime`	主代理上最后一个处理的事件的 TCO 和 TC 之间的估计时间。
`ReplicaWaitTime`	副本代理上最后一个处理的事件的 TCO 和 TP 之间的估计时间。
`QueueSize`	主代理上复制队列中未确认的事件总数。

`TotalReplicationLag` 和 `ReplicationLag` 描述主代理和副本代理之间的延迟复制。这两个指标还可用于估计完成正在进行的切换或失效转移操作所需的时间。

`PrimaryWaitTime` 和 `ReplicaWaitTime` 可用于确定复制过程中正在发生的任何问题。如果此指标的值持续增长，则可能表明复制过程已降级或暂停。由于网络分区、代理启动和恢复时间长等问题，可能会导致复制缓慢。

Amazon MQ for ActiveMQ 中的限额

本主题列出了 Amazon MQ 中的配额。可以为特定 AWS 账户更改以下许多配额。要请求提高限制，请参阅《Amazon Web Services 一般参考》中的 AWS 服务限额。即使提高限制后，更新后的限制也将不可见。有关查看亚马逊当前连接限制的更多信息 CloudWatch，请参阅使用亚马逊监控亚马逊 MQ 代理。 CloudWatch

> **Note**
>
> 有关 Amazon MQ for RabbitMQ 的限额，请参阅 Amazon MQ for RabbitMQ 中的限额。

主题

代理

下表列出了与 Amazon MQ for ActiveMQ 代理程序相关的限额。

限制	描述
代理名称	• 在经纪商区域和您的 AWS 账户中必须是唯一的。 • 长度必须介于 1 到 50 个字符之间。 • 只能包含 ASCII 可打印字符集中指定的字符。 • 只能包含字母数字字符、短划线、句点、下划线和波浪线 (- . _ ~)。

限制	描述
每个区域的代理数	50
小型代理每个协议的线级连接数	`mq.*.micro` 实例类型代理为 300。
大型代理每个协议的线级连接数	`mq.*.*large` 实例类型代理为 2000。
网络连接器的数量	20
每个代理的安全组	5
中监视的 ActiveMQ 目标（队列和主题）CloudWatch	CloudWatch 仅监控前 1000 个目的地。
在 RabbitMQ 中监视的目的地（队列）CloudWatch	CloudWatch 仅监控前 500 个目的地，按消费者数量排序。
每个代理的标签数	50

配置

下表列出了与 Amazon MQ for ActiveMQ 配置相关的限额。

限制	描述
配置名称	• 长度必须介于 1 到 150 个字符之间。 • 只能包含 ASCII 可打印字符集中指定的字符。 • 只能包含字母数字字符、短划线、句点、下划线和波浪线 (- . _ ~)。
每个配置的修订	300

用户

下表列出了与 Amazon MQ for ActiveMQ 代理程序用户相关的限额。

限制	描述
用户名	• 长度必须介于 1 到 100 个字符之间。 • 只能包含 ASCII 可打印字符集中指定的字符。 • 只能包含字母数字字符、短划线、句点、下划线和波浪线 (- . _ ~)。 • 不能包含逗号 (,)。
密码	• 长度必须介于 12 到 250 个字符之间。 • 只能包含 ASCII 可打印字符集中指定的字符。 • 必须至少包含 4 个唯一字符。 • 不能包含逗号 (,)。
每个代理的用户（简单身份验证）	250
每个用户的组数（简单身份验证）	20

数据存储

下表列出了与 Amazon MQ for ActiveMQ 数据存储相关的限额。

限制	描述
每个小型代理的存储容量	`mq.*.micro` 实例类型代理为 20GB。有关 Amazon MQ 实例类型的更多信息，请参阅Broker instance types。
每个大型代理的存储容量	`mq.*.*large` 实例类型代理为 200GB。有关 Amazon MQ 实例类型的更多信息，请参阅Broker instance types。

限制	描述
Amazon EBS 支持的每个代理的任务计划程序使用限制	50GB。有关任务计划程序使用的更多信息，请参阅《Apache ActiveMQ API 文档》中的`JobSchedulerUsage`。
每个小型代理的临时存储容量	`mq.*.micro` 实例类型代理为 5GB。
每个大型代理的临时存储容量	`mq.*.*large` 实例类型代理为 50GB。

API 限制

以下限制配额是在所有 Amazon MQ API 中按 AWS 账户汇总的，以保持服务带宽。要了解有关 Amazon MQ API 的更多信息，请参阅 Amazon MQ REST API 参考。

> **Important**
>
> 这些配额不适用于 Amazon MQ for ActiveMQ 或 Amazon MQ for RabbitMQ 代理消息 API。例如，Amazon MQ 不会限制消息的发送或接收。

API 突增限制	API 速率限制
100	15

使用 Amazon MQ for RabbitMQ

利用 Amazon MQ，可以轻松使用适合您的需求的计算和存储资源创建消息代理。您可以使用AWS Management Console、Amazon MQ REST API 或 AWS Command Line Interface 创建、管理和删除代理。

这部分将介绍 ActiveMQ 和 RabbitMQ 引擎类型的消息代理的基本要素，列出可用的 Amazon MQ 代理实例类型及其状态，并概述代理架构和配置选项。

要了解有关 Amazon MQ REST API 的信息，请参阅 Amazon MQ REST API 参考。

主题

RabbitMQ 引擎

本节介绍 RabbitMQ 代理的基本元素及其支持的插件，并概述 Amazon MQ 上的 RabbitMQ 代理架构选项。

主题

基本元素

本部分介绍对理解 RabbitMQ on Amazon MQ 必不可少的主要概念。

主题

- 代理默认设置
- 代理实例类型
- 亚马逊 MQ for RabbitMQ 尺码指南
- 配置
- 用户
- 插件
- 策略

代理

代理 是运行在 Amazon MQ 上的消息代理环境。它是 Amazon MQ 的基本构建块。代理实例类（m5、t3）和大小（large、micro）的综合描述是一个代理实例类型（例如 mq.m5.large)。有关更多信息，请参阅 Broker instance types。

- 单实例代理由位于网络负载均衡器（NLB）后面的一个可用区中的一个代理组成。代理可与应用程序和 Amazon EBS 存储卷进行通信。
- 集群部署是网络负载均衡器后面的三个 RabbitMQ 代理节点的逻辑分组，每个节点共享用户、队列和跨多个可用区（AZ）的分布式状态。

有关更多信息，请参阅 代理架构。

您可以启用自动次要版本升级 以在 RabbitMQ 引擎的新版本发布时自动升级到代理引擎的新次要版本。自动升级在维护时段内发生，该维护时段使用星期几、几点（24 小时格式）和时区（默认为 UTC）定义。

支持的协议

您可以访问您的 RabbitMQ 代理，方法是使用 RabbitMQ 支持的任何编程语言并通过为以下协议启用 TLS：

- AMQP (0-9-1)

侦听器端口

Amazon MQ 托管式 RabbitMQ 代理支持以下侦听器端口，以实现通过amqps进行应用程序层面的连接，以及使用 RabbitMQ Web 控制台和管理 API 进行客户端连接。

- 侦听器端口 5671 – 用于通过安全 AMQP URL 建立的连接。例如，提供一个代理 ID 为 b-c8352341-ec91-4a78-ad9c-a43f23d325bb 的代理，并且部署在 us-west-2 区域中，以下是该代理的完整 amqp URL：b-c8352341-ec91-4a78-ad9c-a43f23d325bb.mq.us-west-2.amazonaws.com:5671。
- 侦听器端口 443 和 15671 – 两个侦听器端口可以互换使用，通过 RabbitMQ Web 控制台或管理 API 访问代理。

Attributes

RabbitMQ 代理具有几个属性：

- 名称。例如，MyBroker。
- ID。例如，b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9。
- Amazon Resource Name（ARN）。例如，arn:aws:mq:us-east-2:123456789012:broker:MyBroker:b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9。
- RabbitMQ Web 控制台 URL。例如，https://b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-east-2.amazonaws.com。

  有关更多信息，请参阅 RabbitMQ 文档中的 RabbitMQ Web 控制台。
- 安全的 AMQP 端点。例如，amqps://b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-east-2.amazonaws.com。

有关代理属性的完整列表，请参阅《Amazon MQ REST API 参考》中的以下内容：

- REST 操作 ID：Broker
- REST 操作 ID：Brokers
- REST 操作 ID：Broker Reboot

代理默认设置

当您为 RabbitMQ 代理创建 Amazon MQ 时，Amazon MQ 会应用一组默认的代理策略和虚拟主机限制来优化代理的性能。Amazon MQ 仅将虚拟主机限制应用于默认的（/）虚拟主机。Amazon MQ 不会将默认策略应用于新创建的虚拟主机。建议为所有新的和现有的代理保留这些默认值。但是，您可以随时修改、覆盖或删除这些默认设置。

Amazon MQ 根据您在创建代理时选择的实例类型和代理部署模式创建策略和限制。默认策略根据部署模式命名，如下所示：

- 单实例 – AWS-DEFAULT-POLICY-SINGLE-INSTANCE
- 集群部署 – AWS-DEFAULT-POLICY-CLUSTER-MULTI-AZ

对于单实例代理，Amazon MQ 会将策略优先级值设置为 0。要覆盖默认优先级值，可以创建具有较高优先级值的自定义策略。对于集群部署，Amazon MQ 将优先级值设置为代理默认值 1。要为集群创建您自己的自定义策略，请分配一个大于 1 的优先级值。

> **Note**
>
> 在集群部署中，经典镜像和高可用性（HA）需要 ha-mode 和 ha-sync-mode 代理策略。如果删除默认 AWS-DEFAULT-POLICY-CLUSTER-MULTI-AZ 策略，Amazon MQ 将使用 ha-all-AWS-OWNED-DO-NOT-DELETE 策略，优先级值为 0。这可以确保所需的 ha-mode 和 ha-sync-mode 策略仍然有效。如果您创建自己的自定义策略，Amazon MQ 自动将 ha-mode 和 ha-sync-mode 添加到您的策略定义。

主题

策略和限制说明

以下列表描述了 Amazon MQ 应用于新创建的代理的默认策略和限制。max-length、max-queues 和 max-connections 的值因代理的实例类型和部署模式而异。这些值列于建议的默认值部分。

- **queue-mode: lazy**（策略）– 启用延迟队列。默认情况下，队列会在内存中保留消息缓存，从而使代理能够尽快将消息传递给使用者。这可能会导致代理内存不足并引发高内存警报。延迟队列尝试在可行的情况下尽早将消息移动到磁盘。这意味着在正常操作条件下保留在内存中的消息更少。使用延迟队列，Amazon MQ for RabbitMQ 可以支持更大的邮件负载和更长的队列。请注意，对于某些用例，具有延迟队列的代理可能会稍慢一些。这是因为消息从磁盘移动到代理，而不是从内存中的缓存传递消息。

> 部署模式
>
> 单实例、集群

- **max-length:** ***number-of-messages***（策略）– 设置队列中的消息数量的限制。在集群部署中，该限制可防止在代理重启或维护时段之后暂停队列同步。

> 部署模式
>
> 集群

- **overflow: reject-publish**（策略）– 使用 max-length 策略强制队列在队列中的消息数达到 max-length 值后拒绝新消息。为了确保队列处于溢出状态时消息不会丢失，向代理发布消息的客户端应用程序必须实施发布者确认。有关实施发布者确认的信息，请参阅 RabbitMQ 网站上的发布者确认。

> 部署模式
>
> 集群

- **max-queues:** ***number-of-queues-per-vhost***（虚拟主机限制）– 设置代理中队列数的限制。与max-length策略定义类似，限制集群部署中的队列数量可防止在代理重新启动或维护时段后暂停队列同步。限制队列还可防止过多使用 CPU 来维护队列。

> 部署模式
>
> 单实例、集群

- **max-connections:** ***number-of-connections-per-vhost***（虚拟主机限制）– 设置连接代理的客户端数量限制。根据建议的值限制连接数可防止使用过多的代理内存，以免导致代理产生高内存警报和暂停操作。

> 部署模式
>
> 单实例、集群

建议的默认值

> **Note**
>
> `max-length` 和 `max-queue` 默认限制将根据平均邮件大小 5KB 进行测试和评估。如果您的消息大于 5KB，则需要调整和减少 `max-length` 和 `max-queue` 限制。

下表列出了新创建的代理的默认限制值。Amazon MQ 根据代理的实例类型和部署模式应用这些值。

实例类型	Deployment mode (部署模式)	**max-length**	**max-queues**	**max-connections**
t3.micro	单实例	不适用	500	500
m5.large	单实例	不适用	20000	4,000
	集群	80000,000	4,000	15000
m5.xlarge	单实例	不适用	30000	8000
	集群	90000,000	5000	20000
m5.2xlarge	单实例	不适用	60000	15000
	集群	10,000,000	6000	40000
m5.4xlarge	单实例	不适用	15万	30000
	集群	12,000,000	10000	100000

代理实例类型

> **Important**
>
> 您不能将代理从 `mq.m5.` 实例类型降级为 `mq.t3.micro` 实例类型。

实例类型	vCPU	内存 (GiB)	网络性能	应用场景
mq.t3.micro	2	1	低	评估 **Important** mq.t3.micro 实例类型不支持集群部署。
mq.m5.large	2	8	高	生产
mq.m5.xlarge	4	16	高	生产
mq.m5.2xlarge	8	32	高	
mq.m5.4xlarge	16	64	高	

亚马逊 MQ for RabbitMQ 尺码指南

您可以选择最能支持您的应用程序的代理实例类型。在选择实例类型时，重要的是要考虑会影响代理性能的因素：

- 客户机和队列的数量
- 发送的消息量
- 消息保存在内存中
- 冗余消息

较小的代理实例类型可用于测试应用程序性能。较大的代理实例类型可以处理客户端和队列的生产级别、高吞吐量、内存中的消息和冗余消息。

请务必测试您的代理，以确定适合您的工作负载消息传送要求的实例类型和大小。使用以下规模调整指南来确定最适合您的应用程序的实例类型：

实例类型	Deployment mode (部署模式)	最大连接数	最大通道数
t3.micro	单一实例	500	1500
m5.large	单一实例	5000	15000
	集群	15000	45000
m5.xlarge	单一实例	10000	30000
	集群	30000	90000
m5.2xlarge	单一实例	20000	60000
	集群	60000	180,000
m5.4xlarge	单一实例	40000	120,000
	集群	120,000	360,000

超过连接或频道限制时，将返回以下错误消息。

Channel

```
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"
```

配置

配置中包含您的 RabbitMQ 代理的所有设置（采用 Cuttlefish 格式）。您可以先创建配置，然后创建代理。之后，您可以将配置应用于一个或多个代理

> **Important**
>
> 对配置进行更改不会立即将更改应用到代理。要应用更改，必须等待下一维护时段或者重启代理。有关更多信息，请参阅 Amazon MQ 代理配置生命周期。
> 目前，您无法删除配置。

有关创建、编辑和管理配置的信息，请参阅以下内容：

- Creating and applying broker configurations
- RabbitMQ Broker Configurations

要跟踪您对配置所做的更改，可以创建配置版本。有关更多信息，请参阅 Creating and applying broker configurations。

Attributes

代理配置具有几个属性，例如：

- 名称 (`MyConfiguration`)
- ID (`c-1234a5b6-78cd-901e-2fgh-3i45j6k178l9`)
- Amazon Resource Name (ARN) (`arn:aws:mq:us-east-2:123456789012:configuration:c-1234a5b6-78cd-901e-2fgh-3i45j6k178l9`)

有关配置属性的完整列表，请参阅《Amazon MQ REST API 参考》中的以下内容：

- REST 操作 ID：Configuration
- REST 操作 ID：Configurations

有关配置修订属性的完整列表，请参阅以下内容：

- REST 操作 ID：Configuration Revision
- REST 操作 ID：Configuration Revisions

用户

每个 AMQP 0-9-1 客户端连接都有一个必须进行身份验证的关联用户。每个客户端连接还以相应虚拟主机（vhost）为目标，用户必须拥有其一组权限。用户可能有权配置、写入和读取虚拟主机中的队列和交换器。在建立连接时指定用户凭证和目标虚拟主机。

当您首次创建 Amazon MQ for RabbitMQ 代理程序时，Amazon MQ 使用您提供的登录凭证创建带有 `administrator` 标签的 RabbitMQ 用户。然后，您可以通过 RabbitMQ 管理 API 或 RabbitMQ Web 控制台添加和管理用户。您还可以使用 RabbitMQ Web 控制台或管理 API 来设置或修改用户权限和标签。

> **Note**
>
> 不通过 Amazon MQ 用户 API 存储或显示 RabbitMQ 用户。

> **Important**
>
> 适用于 RabbitMQ 的 Amazon MQ 不支持用户名 “访客”，并且将在您创建新经纪人时删除默认访客账户。Amazon MQ 还将定期删除任何由客户创建的名为 “访客” 的账户。

要使用 RabbitMQ 管理 API 创建新用户，请使用以下 API 终端节点和请求体。将 *`username`* 和 *`password`* 替换为新的登录凭证。

```
PUT /api/users/username HTTP/1.1

            {"password":"password","tags":"administrator"}
```

> **Important**
>
> - 请勿在代理用户名中添加个人身份信息（PII）或其他机密或敏感信息。其他 AWS 服务（包括 CloudWatch 日志）可以访问经纪人的用户名。代理用户名不适合用于私有或敏感数据。
> - 如果您忘记了在创建代理时设置的管理员密码，则无法重置凭证。如果您创建了多个管理员，则可以使用其他管理员用户登录，然后重置或重新创建您的凭证。如果您只有一个管理员用户，则必须删除代理并使用新凭证创建一个新代理。我们建议在删除代理之前使用或备份消息。

tags 键是必需的，并且是用逗号分隔的用户标签列表。Amazon MQ 支持 administrator、management、monitoring 和 policymaker 用户标签。

您可以使用以下 API 终端节点和请求体为单个用户设置权限。将 *vhost* 和 *username* 替换为您的信息。对于默认虚拟主机 /，使用 %2F。

```
PUT /api/permissions/vhost/username HTTP/1.1

            {"configure":".*","write":".*","read":".*"}
```

> **Note**
>
> configure、read 和 write 键都是必需的。

通过使用通配符 .* 值，则此操作将向用户授予指定虚拟主机中所有队列的读取、写入和配置权限。有关通过 RabbitMQ 管理 API 管理用户的更多信息，请参阅 RabbitMQ 管理 HTTP API。

插件

Amazon MQ for RabbitMQ 支持 RabbitMQ 管理插件，该插件为管理 API 和 RabbitMQ Web 控制台提供支持。您可以使用 Web 控制台和管理 API 创建和管理代理用户和策略。

除了管理插件之外，Amazon MQ for RabbitMQ 还支持以下插件。

主题

- Shovel 插件
- 联合插件
- 一致性哈希交换器插件

Shovel 插件

Amazon MQ 托管式代理支持 RabbitMQ shovel，允许您将消息从一个代理实例上的队列和交换器移动到另一个。您可以使用 shovel 连接松耦合的代理，并将消息从消息负载较重的节点分发出去。

Amazon MQ 托管式 RabbitMQ 代理支持动态 shovel。动态 shovel 使用运行时参数进行配置，并且可以随时通过客户端连接以编程方式启动和停止。例如，使用 RabbitMQ 管理 API，您可以创建 PUT 请

求并发送到以下 API 终端节点来配置动态 shovel。在该示例中，{vhost} 可以替换为代理的虚拟主机的名称，{name} 替换为新的动态 shovel 的名称。

```
/api/parameters/shovel/{vhost}/{name}
```

在请求体中，您必须指定队列或交换器，但不能同时指定两者。下面的这个示例在 src-queue 中指定的本地队列和在 dest-queue 中定义的远程队列之间配置动态 shovel。同样，您可以使用 src-exchange 和 dest-exchange 参数来配置两个交换器之间的 shovel。

```
{
                "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

如果 shovel 目标是私有代理，则无法在队列或交换器之间配置 shovel。只能在公有代理中的队列或交换器之间，或在私有代理中的源与公有代理中的目标之间配置 shovel。

有关使用动态 shovel 的更多信息，请参阅 RabbitMQ 动态 shovel 插件。

Note

Amazon MQ 不支持使用静态 shovel。

联合插件

Amazon MQ 支持联合交换器和队列。通过联合，您可以在各个代理上的队列、交换器和消费者之间复制消息流。联合队列和交易所使用 point-to-point 链接连接到其他经纪商中的对等体。默认情况下，联合交换器只路由一次消息，而联合队列可以根据使用者的需要多次移动消息。

您可以使用联合允许下游代理使用来自上游的交换器或队列的消息。您可以使用 RabbitMQ Web 控制台或管理 API 在下游代理上启用联合。

> **Important**
>
> 如果上游队列或交换器位于私有代理中，则无法配置联合身份验证。只能在公有代理中的队列或交换器之间，或在公有代理中的上游队列或交换器与私有代理中的下游队列或交换器之间配置联合身份验证。

例如，通过管理 API，您可以执行以下操作来配置联合。

- 配置一个或多个定义到其他节点的联合连接的上游。您可以使用 RabbitMQ Web 控制台或管理 API 定义联合连接。使用管理 API，您可以创建 POST 请求并发送到具有以下请求体的 `/api/parameters/federation-upstream/%2f/my-upstream`。

  ```
  {"value":{"uri":"amqp://server-name","expires":3600000}}
  ```

- 配置策略以使您的队列或交换器联合。您可以使用 RabbitMQ Web 控制台或管理 API 配置策略。使用管理 API，您可以创建 POST 请求并发送到具有以下请求体的 `/api/policies/%2f/federate-me`。

  ```
  {"pattern":"^amq\.", "definition":{"federation-upstream-set":"all"}, "apply-to":"exchanges"}
  ```

  > **Note**
  >
  > 请求体假定服务器上的交换器以 amq 开头。使用正则表达式 `^amq\.` 将确保所有名称以“amq”开头的交换器都已启用联合。您的 RabbitMQ 服务器上的交换器可以不同的名称命名。

有关配置联合插件的更多信息，请参阅 RabbitMQ 联合插件。

一致性哈希交换器插件

默认情况下，Amazon MQ for RabbitMQ 支持一致性哈希交换器类型插件。一致性哈希交换器根据通过消息的路由键计算的哈希值将消息路由到队列。如果提供合理均匀的路由密钥，一致性哈希交换可以在队列之间合理均匀地分发消息。

对于绑定到一致哈希交换的队列，绑定密钥用于确定每个队列的绑定权重。 number-as-a-string 具有较高绑定权重的队列将从与之绑定的一致性哈希交换中接收分配比例更高的消息。在一致性哈希交换拓扑中，发布者可以简单地将消息发布到交换中，但必须将使用者明确配置为使用来自特定队列的消息。

有关一致哈希交换的更多信息，请参阅网站上的 RabbitMQ 一致哈希交换类型。 GitHub

策略

您可以使用 Amazon MQ 推荐的默认值来应用自定义策略和限制。如果已删除建议的默认策略和限制，并希望重新创建这些策略和限制，或者创建了其他虚拟主机并希望将默认策略和限制应用于新虚拟主机，则可以使用以下步骤。

Important

要执行下列步骤，您必须拥有具有管理员权限的 Amazon MQ RabbitMQ 代理用户。您可以使用第一次创建代理时创建的管理员用户，也可以使用之后可能创建的其他用户。下表提供了作为正则表达式（正则表达式）模式所需的管理员用户标签和权限。

标签	读取正则表达式	配置正则表达式	写入正则表达式
`administrator`	`.*`	`.*`	`.*`

有关创建 RabbitMQ 用户和管理用户标记和权限的更多信息，请参阅用户。

使用 RabbitMQ Web 控制台应用默认策略和虚拟主机限制

1. 登录 Amazon MQ 控制台。
2. 在左侧导航窗格中，选择 Brokers (代理)。
3. 从代理列表中，选择要向其应用新策略的代理的名称。
4. 在代理详细信息页面的 Connections (连接) 部分，选择 RabbitMQ web console (RabbitMQ Web 控制台) URL。RabbitMQ Web 控制台可在新的浏览器选项卡或窗口中打开。
5. 使用您的代理管理员的用户名和密码登录 RabbitMQ Web 控制台。
6. 在 RabbitMQ Web 控制台页面顶部选择 Admin (管理员)。
7. 在 Admin (管理员) 页面的右侧导航窗格中，选择 Policies (策略)。

8. 在 Policies (策略) 页面上，您可以看到代理现有的 User policies (用户策略) 列表。在 User policies (用户策略) 下，展开 Add / update a policy (添加/更新策略)。
9. 要创建新的代理策略，请在 Add / update a policy (添加/更新策略) 下，执行以下操作：

   a. 对于 Virtual host (虚拟主机)，请从下拉列表中选择要将策略附加到的虚拟主机的名称。要选择默认虚拟主机，请选择 /。

   > **Note**
   >
   > 如果尚未创建其他虚拟主机，则 RabbitMQ 控制台中不显示 Virtual host (虚拟主机) 选项，并且策略仅应用于默认虚拟主机。

   b. 对于 Name (名称)，请为您的策略输入名称，例如 **`policy-defaults`**。

   c. 在 Pattern (模式) 中，输入正则表达式模式 **`.*`**，以便策略匹配代理上的所有队列。

   d. 对于 Apply to (应用于)，从下拉列表中选择 Exchanges and queues (交换器和队列)。

   e. 对于 Priority (优先级)，输入一个大于应用于虚拟主机的所有其他策略的整数。您可以在任何给定时间将一组策略定义应用于 RabbitMQ 队列和交换器。RabbitMQ 选择具有最高优先级值的匹配策略。有关策略优先级以及如何组合策略的更多信息，请参阅 RabbitMQ 服务器文档中的策略。

   f. 对于 Definition (定义)，添加以下键/值对：

   - **`queue-mode=lazy`**。从下拉列表中选择 String (字符串)。
   - **`overflow=reject-publish`**。从下拉列表中选择 String (字符串)。

     > **Note**
     >
     > 不适用于单实例代理。

   - **`max-length=`*`number-of-messages`***。 根据代理*`number-of-messages`*的实例大小和部署模式（例如集群），替换为 Amazon MQ 的推荐值。**`8000000`** `mq.m5.large`从下拉列表中选择 Number (编号）。

     > **Note**
     >
     > 不适用于单实例代理。

   g. 选择 Add / update policy (添加/更新策略)。

10. 确认 User policies (用户策略) 列表中显示新策略。

> **Note**
>
> 对于集群代理，Amazon MQ 会自动应用 `ha-mode: all` 和 `ha-sync-mode: automatic` 策略定义。

11. 从右侧导航窗格中，选择 Limits (限制)。
12. 在 Limits (限制) 页面上，您可以看到代理现有的 Virtual host limits (虚拟主机限制) 列表。在 Virtual host limits (虚拟主机限制) 下，展开 Set / update a virtual host limit (设置/更新虚拟主机限制)。
13. 要创建新的虚拟主机限制，请在 Set / update a virtual host limit (设置/更新虚拟主机限制) 中，执行以下操作：
    a. 对于 Virtual host (虚拟主机)，请从下拉列表中选择要将策略附加到的虚拟主机的名称。要选择默认虚拟主机，请选择 /。
    b. 对于 Limit (限制)，从下拉选项中选择 max-connections。
    c. 对于 Value (值)，根据代理的实例大小和部署模式输入 Amazon MQ 建议值，例如，对于 `mq.m5.large` 集群输入 **`15000`**。
    d. 选择 Set / update limit (设定/更新限制)。
    e. 重复上述步骤，对于 Limit (限制)，从下拉选项中选择 max-queue (最大队列数)。
14. 确认新限制在 Virtual host limits (虚拟主机限制) 列表中显示。

使用 RabbitMQ 管理 API 应用默认策略和虚拟主机限制

1. 登录 Amazon MQ 控制台。
2. 在左侧导航窗格中，选择 Brokers (代理)。
3. 从代理列表中，选择要向其应用新策略的代理的名称。
4. 在代理页面的 Connections (连接) 部分，记下 RabbitMQ web console (RabbitMQ Web 控制台) URL。这是您在 HTTP 请求中使用的代理终端节点。
5. 打开您选择的新终端或命令行窗口。
6. 要创建新的代理策略，请输入以下 `curl` 命令。此命令假定默认 / 虚拟主机上有一个队列，该队列编码为 `%2F`。要将策略应用到另一个虚拟主机，请将 `%2F` 替换为虚拟主机的名称。

> **Note**
>
> 将 *username* 和 *password* 替换为管理员登录凭证。根据代理*number-of-messages*的实例大小和部署模式，替换为 Amazon MQ 推荐值。将 *policy-name* 替换为您的策略名称。将 *broker-endpoint* 替换为您之前备注的 URL。

```
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. 要确认新策略已添加到您的代理的用户策略中，请输入以下 curl 命令以列出所有代理策略。

```
curl -i -u username:password broker-endpoint/api/policies
```

8. 创建新的 max-connections 虚拟主机限制，请输入以下 curl 命令。此命令假定默认 / 虚拟主机上有一个队列，该队列编码为 %2F。要将策略应用到另一个虚拟主机，请将 %2F 替换为虚拟主机的名称。

> **Note**
>
> 将 *username* 和 *password* 替换为管理员登录凭证。根据代理的实例大小和部署模式，将 *max-connections* 替换为 Amazon MQ 建议值。将代理终端节点替换为您之前记下的 URL。

```
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. 要创建新的 max-queues 虚拟主机限制，请重复上一步，但修改 curl 命令，如下所示。

```
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. 要确认新限制已添加到您的代理的虚拟主机限制，请输入以下 `curl` 命令以列出所有代理虚拟主机限制。

```
curl -i -u username:password broker-endpoint/api/vhost-limits
```

代理架构

可以单实例代理或集群部署方式创建 RabbitMQ 代理。对于这两种部署模式，Amazon MQ 通过冗余存储其数据来提供高持久性。

您可以访问您的 RabbitMQ 代理，方法是使用 RabbitMQ 支持的任何编程语言并通过为以下协议启用 TLS：

- AMQP (0-9-1)

主题

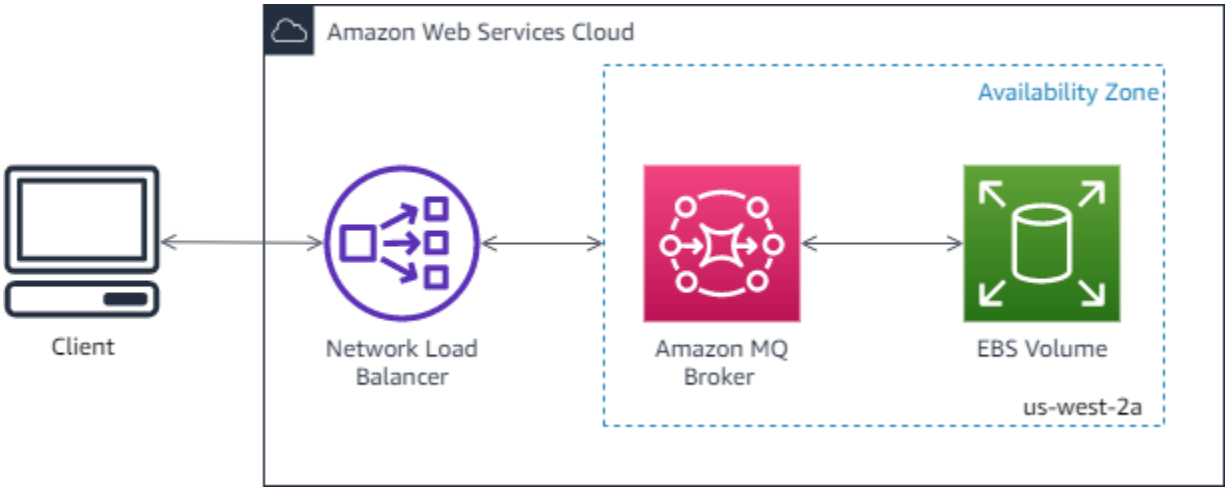
- 单实例代理
- 用于实现高可用性的集群部署

单实例代理

单实例代理由位于网络负载均衡器（NLB）后面的一个可用区中的一个代理组成。代理可与您的应用程序和 Amazon EBS 存储卷进行通信。Amazon EBS 提供针对低延迟和高吞吐量进行了优化的块级存储。

使用网络负载均衡器可确保您的 Amazon MQ for RabbitMQ 代理终端节点在维护时段期间或由于底层 Amazon EC2 硬件故障更换代理实例时保持不变。网络负载均衡器允许您的应用程序和用户继续使用相同的终端节点连接到代理。

下图说明了 Amazon MQ for RabbitMQ 单实例代理。



用于实现高可用性的集群部署

集群部署是网络负载均衡器后面的三个 RabbitMQ 代理节点的逻辑分组，每个节点共享用户、队列和跨多个可用区（AZ）的分布式状态。

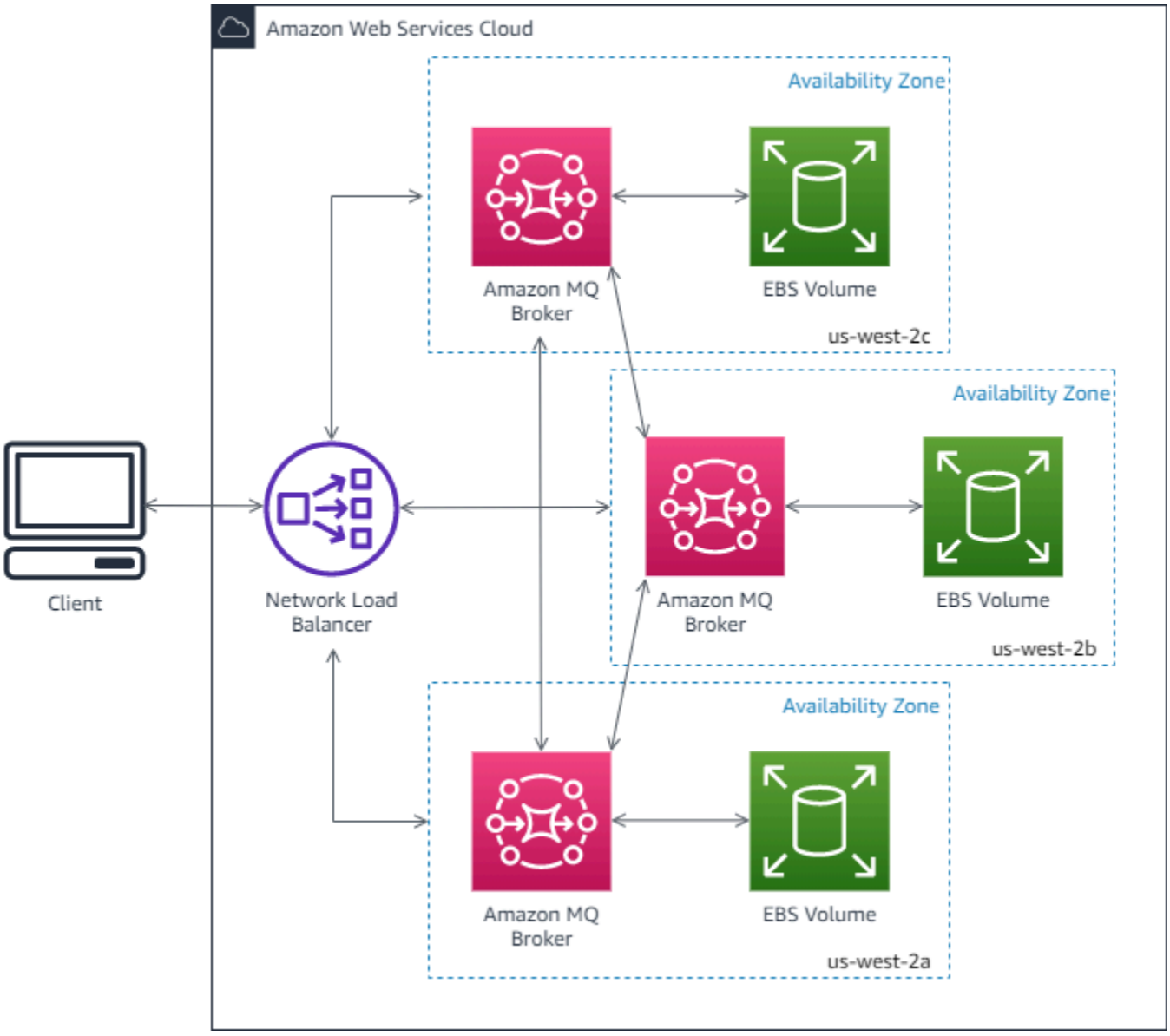
在集群部署中，Amazon MQ 会自动管理代理策略，以在所有节点上启用经典镜像，从而确保高可用性（HA）。每个镜像队列由一个主节点和一个或多个镜像组成。每个队列都有自己的主节点。给定队列的所有操作首先应用于队列的主节点，然后传输到镜像。Amazon MQ 可创建默认系统策略，该策略将 `ha-mode` 设置为 `all`，将 `ha-sync-mode` 设置为 `automatic`。这可确保数据跨不同可用区复制到集群中的所有节点，以获得更好的持久性。

> **Note**
>
> 在维护时段，对集群的所有维护都一次执行一个节点，从而始终保持至少两个运行节点。每次关闭节点时，客户端与该节点的连接都会断开，需要重新建立。您必须确保您的客户端代码设计为自动重新连接到您的集群。有关连接恢复的更多信息，请参阅the section called “自动从网络故障中恢复”。
>
> 因为 Amazon MQ 设置了 `ha-sync-mode: automatic`，在维护时段期间，每当有个节点重新加入集群时，队列都将同步。队列同步会阻止所有其他队列操作。通过保持较短的队列，可以在维护时段期间减轻队列同步的影响。

默认策略不应删除。如果您确实删除了此政策，Amazon MQ 将自动重新创建该政策。Amazon MQ 还将确保 HA 属性应用于您在集群代理上创建的所有其他策略。如果您添加的策略没有 HA 属性，Amazon MQ 将为您添加这些属性。如果您添加具有不同高可用性属性的策略，Amazon MQ 将替换它们。有关传统镜像的更多信息，请参阅经典镜像队列。

> **Important**
>
> Amazon MQ 不支持仲裁队列。启用仲裁队列功能标志并创建仲裁队列将导致数据丢失。

下图说明了 RabbitMQ 集群代理部署，在三个可用区（AZ）中有三个节点，每个节点都有自己的 Amazon EBS 卷和共享状态。Amazon EBS 提供针对低延迟和高吞吐量进行了优化的块级存储。



Amazon MQ for RabbitMQ 代理配置

配置中包含您的 RabbitMQ 代理的所有设置（采用 Cuttlefish 格式）。您可以先创建配置，然后创建代理。之后，您可以将配置应用于一个或多个代理。

主题

创建、编辑和应用 RabbitMQ 代理配置

配置中包含您的 RabbitMQ 代理的所有设置（采用 Cuttlefish 格式）。您可以先创建配置，然后创建代理。之后，您可以将配置应用于一个或多个代理

有关更多信息，请参阅下列内容：

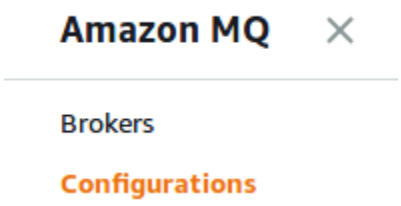
- 配置
- Amazon MQ 代理配置生命周期

以下示例演示如何使用 AWS Management Console创建和应用 RabbitMQ 代理配置。

主题

创建新的配置

1. 登录 Amazon MQ 控制台。
2. 在左侧，展开导航面板，然后选择 Configurations (配置)。



3. 在 Configurations (配置) 页面上，选择 Create configuration (创建配置)。
4. 在 Create configuration（创建配置）页面上的 Details（详细信息）部分中，输入 Configuration name（配置名称）（例如 `MyConfiguration`）并选择 Broker engine（代理引擎）版本。

要了解有关 Amazon MQ for RabbitMQ 支持的 RabbitMQ 引擎版本的更多信息，请参阅the section called “版本管理”。

5. 选择创建配置。

创建新的配置修订

1. 从配置列表中选择***MyConfiguration***。

> **Note**
>
> 始终会在 Amazon MQ 创建配置时为您创建第一个配置修订。

在该***MyConfiguration***页面上，将显示您的新配置修订版使用的代理引擎类型和版本（例如 RabbitMQ 3.xx.xx）。

2. 在配置详细信息选项卡上，会显示配置修订号、描述和 Cuttlefish 格式的代理配置。

> **Note**
>
> 编辑当前配置会创建一个新的配置修订。

3. 选择编辑配置并对 Cuttlefish 配置进行更改。
4. 选择保存。

Save revision (保存修订) 对话框出现。

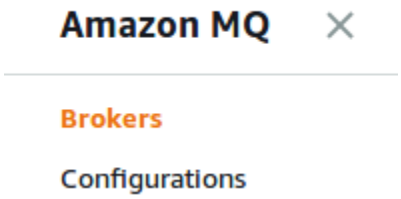
5. （可选）类型 `A description of the changes in this revision`。
6. 选择保存。

将会保存配置的新修订。

> **Important**
>
> 对配置进行更改不会立即将更改应用到代理。要应用更改，必须等待下一维护时段或者重启代理。有关更多信息，请参阅 Amazon MQ 代理配置生命周期。
> 目前，您无法删除配置。

将配置修订应用到代理

1. 在左侧，展开导航面板，然后选择 Brokers (代理)。



2. 从经纪人列表中，选择您的经纪商（例如 MyBroker），然后选择编辑。
3. 在 “编辑 ***MyBroker***” 页面的 “配置” 部分，选择配置和修订版，然后选择 “计划修改”。
4. 在 Schedule broker modifications (计划代理修改) 部分中，选择是在 During the next scheduled maintenance window (下一个计划维护时段期间) 还是 Immediately (立即) 应用修改。

> **Important**
> 您的代理将在重启时脱机。

5. 选择 应用。

   您的配置修订将在指定的时间应用到您的代理。

编辑配置修订

1. 登录 Amazon MQ 控制台。
2. 从经纪人列表中，选择您的经纪商（例如 MyBroker），然后选择编辑。
3. 在***MyBroker***页面上，选择 “编辑”。
4. 在 “编辑 ***MyBroker***” 页面的 “配置” 部分，选择一个配置和一个修订版，然后选择编辑。

> **Note**
> 除非您在创建代理时选择配置，否则会在 Amazon MQ 创建代理时为您创建第一个配置修订。

   在该***MyBroker***页面上，将显示配置使用的代理引擎类型和版本（例如 RabbitMQ 3.xx.xx）。
5. 在配置详细信息选项卡上，会显示配置修订号、描述和 Cuttlefish 格式的代理配置。

> **Note**
>
> 编辑当前配置会创建一个新的配置修订。

6. 选择编辑配置并对 Cuttlefish 配置进行更改。
7. 选择保存。

   Save revision (保存修订) 对话框出现。
8. （可选）类型 `A description of the changes in this revision`。
9. 选择保存。

   将会保存配置的新修订。

> **Important**
>
> 对配置进行更改不会立即将更改应用到代理。要应用更改，必须等待下一维护时段或者重启代理。有关更多信息，请参阅 Amazon MQ 代理配置生命周期。
> 目前，您无法删除配置。

RabbitMQ 配置策略

Amazon MQ for RabbitMQ 现在支持为您的 RabbitMQ 代理创建和应用配置。每个虚拟主机上的原定设置操作员策略具有以下推荐的 HA 属性：

```
name: default_operator_policy_AWS_managed
pattern: .*
apply-to: all
priority: 0
definition: {
ha-mode: all
ha-sync-mode: automatic
}
```

默认情况下，无法通过 AWS Management Console 或管理 API 更改运营商政策。您可以通过在代理配置中添加以下行来启用更改：

```
management.restrictions.operator_policy_changes.disabled=false
```

如果您进行此更改，强烈建议您在自己的操作员策略中包含 HA 属性。有关向代理添加配置的更多信息，请参阅Creating and applying broker configurations。

管理 Amazon MQ for RabbitMQ 引擎版本

RabbitMQ 根据语义版本控制规范将版本号整理为 X.Y.Z。在适用于 RabbitMQ 实现的 Amazon MQ 中，X表示主要版本，Y代表次要版本，表示补丁版本号。Z如果主要版本号发生变化，Amazon MQ 会将版本更改视为主要版本更改。例如，从版本 3. 13 升级到 4. 0 被视为主要版本升级。如果只有次要版本号或补丁版本号发生更改，则版本更改被视为次要更改。例如，从版本 3 升级。 11 .28 到 3。 12 .13 被视为次要版本升级。

亚马逊 MQ for RabbitMQ 建议所有经纪商使用最新支持的次要版本。有关如何升级您的代理引擎版本的说明，请参阅升级 Amazon MQ 代理引擎版本。

> **Important**
>
> Amazon MQ 不支持仲裁队列或流。启用这些功能标志和创建仲裁队列或流将导致数据丢失。
> Amazon MQ 不支持 RabbitMQ 3.9 中推出的在 JSON 中使用结构化日志记录

亚马逊 MQ 上支持的引擎版本 RabbitMQ

Amazon MQ 版本支持日历会显示代理引擎版本何时终止支持。当某个版本的支持终止时，Amazon MQ 会自动将该版本上的所有代理升级到下一个支持的版本。在版本终止支持之前，Amazon MQ 会至少在 90 天内发出通知。

RabbitMQ 版本	亚马逊 MQ 的支持已终止
3.12（推荐）	
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

创建新的 Amazon MQ for RabbitMQ 代理时，您可以指定任何支持的 RabbitMQ 引擎版本。如果您使用创建代理，Amazon MQ 会自动默认为最新的引擎版本号。 AWS Management Console 如果您使用 AWS CLI 或 Amazon MQ API 创建代理，则需要引擎版本号。如果不提供版本号，则操作会导致异常。要了解更多信息，请参阅《AWS CLI 命令参考》中的`create-broker`和《Amazon MQ REST API 参考》中的`CreateBroker`。

引擎版本升级

您可以随时手动将您的代理升级到下一个支持的主要版本、次要版本或补丁版本。当您开启自动次要版本升级时，Amazon MQ 将在维护时段内将您的代理升级到支持的最新补丁版本。

有关手动升级经纪商的更多信息，请参阅the section called “升级引擎版本”。

> **Important**
>
> RabbitMQ 仅允许增量版本更新（例如：3.9.x 到 3.10.x）。更新时不能跳过次要版本（例如：3.8.x 到 3.11.x）。

单实例代理程序在重启时将处于脱机状态。对于集群代理，镜像队列必须在重启期间同步。队列越长，队列同步过程可能需要更长的时间。在队列同步过程中，使用者和生产者无法使用队列。队列同步过程完成后，代理将再次可用。为了最大限度地减少影响，我们建议在流量较低的时段进行升级。有关版本升级最佳做法的更多信息，请参阅Amazon MQ for RabbitMQ 最佳实践。

列出支持的引擎版本

您可以使用`describe-broker-instance-options` AWS CLI 命令列出所有支持的次要和主要引擎版本。

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

要按引擎和实例类型筛选结果，请使用`--engine-type`和`--host-instance-type`选项，如以下所示。

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

例如，要筛选 RabbitMQ 和 `mq.m5.large` 实例类型的结果，请将 *`engine-type`* 替换为 `RABBITMQ`，并将 *`instance-type`* 替换为 `mq.m5.large`。

RabbitMQ 教程

以下教程展示如何在 Amazon MQ 上配置和使用 RabbitMQ。要了解有关使用各种编程语言（如 Node.js、Python、.NET 等）支持的客户端库的更多信息，请参阅《RabbitMQ 入门指南》中的 RabbitMQ 教程。

主题

编辑代理首选项

您可以编辑您的代理首选项，例如使用AWS Management Console禁用或启用 CloudWatch Logs。

编辑 RabbitMQ 代理选项

1. 登录 Amazon MQ 控制台。
2. 从代理列表中，选择您的代理（例如 MyBroker），然后选择 Edit (编辑)。
3. 在 Edit ***MyBroker*** (编辑 MyBroker) 页面的 Specifications (规范) 部分中，选择 Broker engine version (代理引擎版本) 或 Broker Instance type (代理实例类型)。
4. 在 CloudWatch Logs 部分，单击切换按钮以启用或禁用常规日志。没有必须完成的其他步骤。

   > **Note**
   >
   > - 对于 RabbitMQ 代理，Amazon MQ 自动使用服务相关角色（SLR）将常规日志发布到 CloudWatch。有关更多信息，请参阅the section called “使用服务相关角色”
   > - Amazon MQ 不支持 RabbitMQ 代理的审核日志记录。

5. 在 Maintenance (维护) 部分中，配置您的代理的维护计划：

   要在AWS发布新版本时将代理升级到新版本，请选择 Enable automatic minor version upgrades (启用自动次要版本升级)。自动升级在维护时段内发生，该维护时段使用星期几、几点（24 小时格式）和时区（默认为 UTC）定义。
6. 选择 Schedule modifications (计划修改)。

> **Note**
>
> 如果您仅选择 Enable automatic minor version upgrades (启用自动次要版本升级)，该按钮将变为 Save (保存)，因为不必重启代理。

您的首选项将在指定的时间应用到您的代理。

将 Python Pika 与 Amazon MQ for RabbitMQ 结合使用

以下教程说明如何使用 TLS 设置 Python Pika，并将 TLS 配置为连接到 Amazon MQ for RabbitMQ 代理。Pika 是适用于 RabbitMQ 的 AMQP 0-9-1 协议的 Python 实现。本教程可指导您安装 Pika、声明队列、设置发布者以向代理的默认交换器发送消息，以及设置使用者以接收队列中的消息。

主题

先决条件

要完成本教程中的步骤，您需要满足以下先决条件：

- Amazon MQ for RabbitMQ 代理。有关更多信息，请参阅创建 Amazon MQ for RabbitMQ 代理。
- 安装适用于操作系统的 Python 3。
- 使用 Python `pip` 安装 Pika。要安装 Pika，请打开新的终端窗口并运行以下内容。

```
$ python3 -m pip install pika
```

权限

在本教程中，您需要至少一个 Amazon MQ for RabbitMQ 代理用户，该用户具有写入和读取虚拟主机的权限。下表描述了作为正则表达式 (regexp) 模式所需的最小权限。

标签	配置正则表达式	写入正则表达式	读取正则表达式
none		.*	.*

列出的用户权限仅向用户提供读写权限，而不授予在代理上执行管理操作的管理插件访问权限。您可以通过提供限制用户访问指定队列的 regexp 模式来进一步限制权限。例如，如果将读取 regexp 模式更改为 `^[hello world].*`，用户只拥有从以 `hello world` 开头的队列中读取的权限。

有关创建 RabbitMQ 用户和管理用户标记和权限的更多信息，请参阅 用户。

步骤一：创建基本的 Python Pika 客户端

请执行以下操作来创建 Python Pika 客户端基类，该基类定义构造函数，并在与 Amazon MQ for RabbitMQ 代理交互时提供 TLS 配置所需的 SSL 上下文。

1. 打开新的终端窗口，为项目创建新目录，然后导航到该目录。

```
$ mkdir pika-tutorial
$ cd pika-tutorial
```

2. 创建一个名为 `basicClient.py` 的文件，其中包含以下 Python 代码。

```
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()
```

现在，您可以为继承自 BasicPikaClient 的发布者和使用者定义其他类。

步骤二：创建发布者并发送消息

要创建声明队列的发布者并发送单条消息，请执行以下操作。

1. 复制以下代码示例的内容，并将其作为 publisher.py 保存在本地上一个步骤创建的相同目录中。

```
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 类继承自 BasicPikaClient 并实施了用于声明队列、向队列发送消息和关闭连接的其他方法。代码示例使用等于队列名称的路由密钥，将消息路由到默认交换器。

2. 在 `if __name__ == "__main__":` 下，将传递给 BasicMessageSender 构造函数语句的参数替换为以下信息。

   - **<broker-id>** – Amazon MQ 为代理生成的唯一 ID。您可以通过代理 ARN 解析 ID。例如，给定以下 ARN `arn:aws:mq:us-east-2:123456789012:broker:MyBroker:b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9`，代理 ID 将为 `b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9`。
   - **<username>** - 具有足够权限向代理写入消息的代理用户的用户名。
   - **<password>** - 具有足够权限向代理写入消息的代理用户的密码。
   - **<region>** - 您创建 Amazon MQ for RabbitMQ 代理的 AWS 区域。例如，`us-west-2`。

3. 在创建 `publisher.py` 的同一目录中运行以下命令。

```
$ python3 publisher.py
```

如果代码运行成功，您将在终端窗口中看到以下输出。

```
Trying to declare queue(hello world queue)...
Sent message. Exchange: , Routing Key: hello world queue, Body: b'Hello World!'
```

步骤三：创建使用者并接收消息

要创建从队列接收单条消息的使用者，请执行以下操作。

1. 复制以下代码示例的内容，并将其作为 consumer.py 保存在本地同一目录中。

```
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()
```

   与您在上一个步骤中创建的发布者类似，BasicMessageReciever 继承自 BasicPikaClient 并实施用于接收单条消息和关闭连接的其他方法。

2. 在 `if __name__ == "__main__":` 语句下，将传递给 BasicMessageReciever 构造函数的参数替换为您的信息。

3. 在您的项目目录中运行以下命令。

```
$ python3 consumer.py
```

如果代码运行成功，您将在终端窗口中看到消息正文以及包括路由密钥的标头。

```
<Basic.GetOk(['delivery_tag=1', 'exchange=', 'message_count=0',
 'redelivered=False', 'routing_key=hello world queue'])> <BasicProperties> b'Hello
 World!'
```

步骤四：（可选）设置事件循环并使用消息

要使用队列中的多条消息，请使用 Pika 的 basic_consume 方法和回调函数，如下所示

1. 在 consumer.py 中，将以下方法定义添加到 BasicMessageReceiver 类。

```
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. 在 consumer.py 中的 if __name__ == "__main__": 下，调用您在上一个步骤中定义的 consume_messages 方法。

```
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. 再次运行 `consumer.py`，如果运行成功，队列消息将显示在终端窗口中。

```
[*] Waiting for messages. To exit press CTRL+C
[x] Received b'Hello World!'
[x] Received b'Hello World!'
...
```

接下来做什么？

- 有关其他支持的 RabbitMQ 客户端库的更多信息，请参阅 RabbitMQ 网站上的 RabbitMQ 客户端文档。

解决 RabbitMQ 暂停队列同步的问题

在 Amazon MQ for RabbitMQ 集群部署中，发布到每个队列的消息跨三个代理节点进行复制。这种复制称为镜像，为 RabbitMQ 代理提供高可用性（HA）。集群部署中的队列由一个节点上的主副本和一个或多个镜像组成。应用于镜像队列的每个操作（包括入队消息）首先应用于主队列，然后在其镜像之间进行复制。

例如，假设一个跨三个节点复制的镜像队列：主节点（`main`）和两个镜像（`mirror-1` 和 `mirror-2`）。如果此镜像队列中的所有消息都成功传播到所有镜像，则队列将同步。如果某个节点（`mirror-1`）在一段时间内变得不可用，则该队列仍可运行并可继续将消息排入队列。但是，对于要同步的队列，在 `mirror-1` 不可用时发布到 `main` 的消息必须复制到 `mirror-1`。

有关镜像的更多信息，请参阅 RabbitMQ 网站上的经典镜像队列。

维护和队列同步

在维护时段，Amazon MQ 每次执行一个节点的所有维护工作，以确保代理保持正常运行。因此，在每个节点恢复正常运行时，队列可能需要同步。在同步过程中，需要复制到镜像的消息将从相应的 Amazon Elastic Block Store（Amazon EBS）卷加载到内存中，以进行批处理。批处理消息可以让队列更快地同步。

如果队列保持简短且消息较少，则队列会按预期成功同步并恢复正常运行。但是，如果批处理中的数据量接近节点的内存限制，节点会引发高内存警报，暂停队列同步。您可以通过比较 CloudWatch 中的 `RabbitMemUsed` 和 `RabbitMqMemLimit` 代理节点指标来确认内存使用情况。在消耗或删除消息或批处理中的消息数量减少之前，同步无法完成。

> **Note**
>
> 减少队列同步批处理大小可能会导致更多的复制事务。

要解决暂停的队列同步，请按照本教程中的步骤操作，其中演示了应用 `ha-sync-batch-size` 策略和重新启动队列同步的过程。

主题

先决条件

在本教程中，您必须拥有具有管理员权限的 Amazon MQ for RabbitMQ 代理用户。您可以使用第一次创建代理时创建的管理员用户，也可以使用之后可能创建的其他用户。下表提供了作为正则表达式（正则表达式）模式所需的管理员用户标签和权限。

标签	读取正则表达式	配置正则表达式	写入正则表达式
`administrator`	`.*`	`.*`	`.*`

有关创建 RabbitMQ 用户和管理用户标记和权限的更多信息，请参阅用户。

步骤 1：应用 **`ha-sync-batch-size`** 策略

以下过程演示了如何添加适用于代理上所创建的所有队列的策略。您可以使用 RabbitMQ Web 控制台或 RabbitMQ 管理 API。有关更多信息，请参阅 RabbitMQ 网站上的管理插件。

使用 RabbitMQ Web 控制台应用 **ha-sync-batch-size** 策略

1. 登录 Amazon MQ 控制台。
2. 在左侧导航窗格中，选择 Brokers (代理)。
3. 从代理列表中，选择要向其应用新策略的代理的名称。
4. 在代理页面的 Connections (连接) 部分，选择 RabbitMQ web console (RabbitMQ Web 控制台) URL。RabbitMQ Web 控制台可在新的浏览器选项卡或窗口中打开。
5. 使用您的代理程序管理员的登录凭证登录 RabbitMQ Web 控制台。
6. 在 RabbitMQ Web 控制台页面顶部选择 Admin (管理员)。
7. 在 Admin (管理员) 页面的右侧导航窗格中，选择 Policies (策略)。
8. 在 Policies (策略) 页面上，您可以看到代理现有的 User policies (用户策略) 列表。在 User policies (用户策略) 下，展开 Add / update a policy (添加/更新策略)。

   > **Note**
   >
   > 默认情况下，Amazon MQ for RabbitMQ 集群是使用名为 `ha-all-AWS-OWNED-DO-NOT-DELETE` 的初始代理策略创建的。Amazon MQ 管理此策略，以确保代理上的每个队列都复制到所有三个节点，并且队列自动同步。

9. 要创建新的代理策略，请在 Add / update a policy (添加/更新策略) 下，执行以下操作：

   a. 对于 Name (名称)，请为您的策略输入名称，例如 **batch-size-policy**。

   b. 在 Pattern (模式) 中，输入正则表达式模式 **.***，以便策略匹配代理上的所有队列。

   c. 对于 Apply to (应用于)，从下拉列表中选择 Exchanges and queues (交换器和队列)。

   d. 对于 Priority (优先级)，输入一个大于应用于虚拟主机的所有其他策略的整数。您可以在任何给定时间将一组策略定义应用于 RabbitMQ 队列和交换器。RabbitMQ 选择具有最高优先级值的匹配策略。有关策略优先级以及如何组合策略的更多信息，请参阅 RabbitMQ 服务器文档中的策略。

   e. 对于 Definition (定义)，添加以下键/值对：

      - **ha-sync-batch-size**=*100*。从下拉列表中选择 Number (编号) 。

> **Note**
>
> 您可能需要根据队列中未同步消息的数量和大小调整和校准 `ha-sync-batch-size` 的值。

- **`ha-mode=all`**。从下拉列表中选择 String (字符串)。

> **Important**
>
> `ha-mode` 定义是所有与 HA 相关的策略所必需的。忽略它会导致验证失败。

- **`ha-sync-mode=automatic`**。从下拉列表中选择 String (字符串)。

> **Note**
>
> `ha-sync-mode` 定义是所有自定义策略所必需的。如果省略该定义，Amazon MQ 会自动追加定义。

f. 选择 Add / update policy (添加/更新策略)。

10. 确认 User policies (用户策略) 列表中显示新策略。

使用 RabbitMQ 管理 API 应用 **`ha-sync-batch-size`** 策略

1. 登录 Amazon MQ 控制台。
2. 在左侧导航窗格中，选择 Brokers (代理)。
3. 从代理列表中，选择要向其应用新策略的代理的名称。
4. 在代理页面的 Connections (连接) 部分，记下 RabbitMQ web console (RabbitMQ Web 控制台) URL。这是您在 HTTP 请求中使用的代理终端节点。
5. 打开您选择的新终端或命令行窗口。
6. 要创建新的代理策略，请输入以下 `curl` 命令。此命令假定默认 / 虚拟主机上有一个队列，该队列编码为 `%2F`。

> **Note**
>
> 将 *username* 和 *password* 替换为代理程序管理员登录凭证。您可能需要根据队列中未同步消息的数量和大小调整和校准 ha-sync-batch-size 的值（*100*）。将代理终端节点替换为您之前记下的 URL。

```
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. 要确认新策略已添加到您的代理的用户策略中，请输入以下 curl 命令以列出所有代理策略。

```
curl -i -u username:password https://b-589c045f-f8ln-4ab0-a89c-co62e1c32ef8.mq.us-
west-2.amazonaws.com/api/policies
```

步骤 2：重新启动队列同步

将新的 ha-sync-batch-size 策略应用于您的代理后，重新启动队列同步。

使用 RabbitMQ Web 控制台重新启动队列同步

> **Note**
>
> 要打开 RabbitMQ Web 控制台，请参阅本教程第 1 步中的先前说明。

1. 在 RabbitMQ Web 控制台页面顶部选择 Queues (队列)。
2. 在 Queues (队列) 页面上的 All queues (所有队列) 下，找到已暂停的队列。在 Features (功能) 列中，您的队列应列出您创建的新策略的名称（例如 batch-size-policy）。
3. 要重新启动具有较小批处理大小的同步过程，请选择 Restart sync (重新启动同步)。

> **Note**
>
> 如果同步暂停并且未成功完成，请尝试减少 `ha-sync-batch-size` 值并重新启动队列同步。

后续步骤

- 队列成功同步后，您可以通过查看 Amazon CloudWatch 指标 `RabbitMQMemUsed` 来监控 RabbitMQ 节点使用的内存量。您还可以查看 `RabbitMQMemLimit` 指标以监控节点的内存限制。有关更多信息，请参阅访问 Amazon MQ 的 CloudWatch 指标和记录和监控 Amazon MQ for RabbitMQ 代理。
- 为防止队列同步暂停，建议保持队列较短并处理消息。对于消息较大的工作负载，我们还建议您将代理实例类型升级到具有更多内存的更大实例大小。有关代理实例类型和编辑代理首选项的更多信息，请参阅Amazon MQ for RabbitMQ 实例类型和编辑代理首选项。
- 当您创建新的 Amazon MQ for RabbitMQ 时，Amazon MQ 会应用一组默认策略和虚拟主机限制来优化代理性能。如果您的代理没有建议的默认策略和限制，建议您自己创建。有关创建默认策略和虚拟主机限制的更多信息，请参阅the section called “代理默认设置”。

相关资源

- UpdateBrokerInput – 使用此代理属性通过 Amazon MQ API 更新代理实例类型。
- 参数和策略（RabbitMQ 服务器文档）– 在 RabbitMQ 网站上了解有关 RabbitMQ 参数和策略的更多信息。
- RabbitMQ 管理 HTTP API – 了解有关 RabbitMQ 管理 API 的更多信息。

Amazon MQ for RabbitMQ 最佳实践

以此作为参考快速找到在 Amazon MQ 上使用 RabbitMQ 代理最大程度提高性能和降低吞吐量成本的建议。

> **Important**
>
> Amazon MQ 不支持仲裁队列。启用仲裁队列功能标志并创建仲裁队列将导致数据丢失。

> **Important**
>
> 目前，Amazon MQ 不支持流或在 JSON 中使用结构化日志记录（在 RabbitMQ 3.9.x 中推出）。

> **Important**
>
> 适用于 RabbitMQ 的 Amazon MQ 不支持用户名 “访客”，并且将在您创建新经纪人时删除默认访客账户。Amazon MQ 还将定期删除任何由客户创建的名为 “访客” 的账户。

主题

启用延迟队列

如果您正在处理处理大量消息的超长队列，则启用延迟队列可以提高代理性能。

RabbitMQ 的默认行为是在内存中缓存消息，并且仅在代理需要更多可用内存时将其移动到磁盘。将消息从内存移动到磁盘需要时间，并且会停止消息处理。懒人队列通过尽快将消息存储到磁盘来显著加快内存到磁盘的进程，从而减少内存中缓存的消息。

您可以通过在声明时设置 `queue.declare` 参数或通过 RabbitMQ 管理控制台配置策略来启用延迟队列。以下示例演示了如何使用 RabbitMQ Java 客户端库声明延迟队列。

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

```
channel.queueDeclare("myqueue", false, false, false, args);
```

默认情况下，3.12.13 及更高版本上的所有 Amazon MQ for RabbitMQ 队列都表现为延迟队列。要升级到最新版本的 Amazon MQ for RabbitMQ，请参阅。???

> **Note**
>
> 启用延迟队列会增加磁盘输入/输出操作。

使用持久和持续队列

持久消息有助于防止在代理崩溃或重启的情况下丢失数据。持久消息一到达就会立即写入磁盘。但是，与延迟队列不同的是，持久消息同时在内存和磁盘中缓存，除非代理需要更多内存。在需要更多内存的情况下，通过管理将消息存储到磁盘的 RabbitMQ 代理机制从内存中删除消息，通常称为持久性层。

要启用消息持久性，可以将队列声明为 `durable` 并将消息传递模式设置为 `persistent`。以下示例演示了如何使用 RabbitMQ Java 客户端库声明持续队列。

```
boolean durable = true;
channel.queueDeclare("my_queue", durable, false, false, null);
```

将队列配置为持续队列后，您可以通过将 `MessageProperties` 设置为 `PERSISTENT_TEXT_PLAIN` 来将持久消息发送到您的队列，如以下示例所示。

```
import com.rabbitmq.client.MessageProperties;

channel.basicPublish("", "my_queue",
            MessageProperties.PERSISTENT_TEXT_PLAIN,
            message.getBytes());
```

保持队列简短

在集群部署中，包含大量消息的队列可能会导致资源过度利用。当代理被过度利用时，重启 Amazon MQ for RabbitMQ 代理可能会导致性能进一步降低。如果重启，过度利用的代理可能会在 `REBOOT_IN_PROGRESS` 状态下变得反应迟钝。

在维护时段，Amazon MQ 每次执行一个节点的所有维护工作，以确保代理保持正常运行。因此，在每个节点恢复正常运行时，队列可能需要同步。在同步过程中，需要复制到镜像的消息将从相应的

Amazon Elastic Block Store（Amazon EBS）卷加载到内存中，以进行批处理。批处理消息可以让队列更快地同步。

如果队列保持简短且消息较少，则队列会按预期成功同步并恢复正常运行。但是，如果批处理中的数据量接近节点的内存限制，节点会引发高内存警报，暂停队列同步。您可以通过比较中的`RabbitMemUsed`和`RabbitMqMemLimit`代理节点指标来确认内存使用情况 CloudWatch。在消耗或删除消息或批处理中的消息数量减少之前，同步无法完成。

如果集群部署暂停队列同步，我们建议使用或删除消息，以减少队列中的消息数量。一旦队列深度减少且队列同步完成，代理状态将更改为 RUNNING。要解决暂停的队列同步，您还可以应用策略来减少队列同步批处理大小。

> **Warning**
>
> 不要重启资源占用率高的代理。
> 如果在队列同步暂停时重启代理，则代理将重启同步过程，这会在消息从存储传输到节点内存时进一步削弱代理资源，并导致代理在 REBOOT_IN_PROGRESS 状态下变得反应迟钝。

配置确认

当客户端应用程序向代理发回消息传递和使用确认时，它被称为使用者确认。同样，向发布者发送确认的过程称为发布者确认。在使用 RabbitMQ 代理时，确认对于确保数据安全至关重要。

使用者传递确认通常在客户端应用程序上配置。使用 AMQP 0-9-1 时，可以通过配置 `basic.consume` 或使用 `basic.code` 方法获取消息来启用确认。

通常，在通道中启用传递确认。例如，使用 RabbitMQ Java 客户端库时，可以使用 `Channel#basicAck` 来设置一个简单的 `basic.ack` 肯定确认，如以下示例所示。

```
// 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**
>
> 未确认的消息必须在内存中缓存。您可以通过为客户端应用程序配置预提取设置，限制使用者预提取的消息数量。

配置预提取

您可以使用 RabbitMQ 预提取值来优化使用者使用消息的方式。RabbitMQ 通过将预提取计数应用于使用者而不是通道，实现 AMQP 0-9-1 提供的通道预提取机制。预提取值用于指定在任何给定时间向使用者发送的消息数量。默认情况下，RabbitMQ 会为客户端应用程序设置无限制的缓冲区大小。

在为您的 RabbitMQ 使用者设置预提取计数时，需要考虑各种因素。首先，考虑使用者的环境和配置。由于使用者需要在处理消息时将所有消息保存在内存中，因此，较高的预提取值可能会对使用者的性能产生负面影响，在某些情况下，可能会导致使用者同时崩溃。同样，RabbitMQ 代理本身会将其发送的所有消息缓存在内存中，直到收到使用者确认。如果没有为使用者配置自动确认，并且使用者需要相对较长的时间来处理消息，则较高的预提取值可能会导致 RabbitMQ 服务器内存不足。

考虑到上述因素，我们建议始终设置预提取值，以防止由于大量未处理或未确认的消息而导致 RabbitMQ 代理或其使用者出现内存不足的情况。如果您需要优化代理来处理大量消息，您可以使用一系列预提取计数来测试您的代理和使用者，以确定与使用者处理消息所需的时间相比，网络开销在哪个点上变得微不足道。

> **Note**
>
> - 如果您的客户端应用程序已配置为自动确认将消息传递给使用者，则设置预提取值将不起作用。
> - 所有预提取消息都会从队列中删除。

以下示例演示了如何使用 RabbitMQ Java 客户端库为单一使用者设置 `10` 的预提取值。

```
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**
>
> 在 RabbitMQ Java 客户端库中，`global` 标志的默认值设置为 `false`，所以上面的例子可以简单地写成 `channel.basicQos(10)`。

配置 Celery

Python Celery 会发送许多不必要的消息，这些消息会使查找和处理有用的信息变得更加困难。为了降低噪音并使处理更容易，请输入以下命令：

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

自动从网络故障中恢复

我们建议始终启用自动网络恢复，以防止在客户端连接到 RabbitMQ 节点失败的情况下出现严重停机。自版本 `4.0.0` 起，RabbitMQ Java 客户端库默认支持自动网络恢复。

如果在连接的输入/输出循环中引发未处理的异常、检测到套接字读取操作超时，或者如果服务器失去检测信号，则会触发自动连接恢复。

如果客户端和 RabbitMQ 节点之间的初始连接失败，将不会触发自动恢复。我们建议您编写应用程序代码，以便通过重试连接来解决初始连接失败的问题。以下示例演示了如何使用 RabbitMQ Java 客户端库来重试初始网络故障。

```
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**
>
> 如果应用程序使用 `Connection.Close` 方法关闭连接，则不会启用或触发自动网络恢复。

为您的 RabbitMQ 代理启用 Classic Queue v2

我们建议在代理引擎版本 3.10 和 3.11 上启用 Classic Queue v2 (CQv2)，以提高性能，包括：

- 减少内存使用量
- 改善使用者传递
- 提高工作负载的吞吐量，让使用者跟上生产者的步伐

默认情况下，3.12.13 及更高版本的所有亚马逊 MQ for RabbitMQ 队列都使用 CQv2。要升级到最新版本的 Amazon MQ for RabbitMQ，请参阅。???

从 CQv1 迁移到 CQv2

要使用 CQv2，必须先启用`classic_mirrored_queue_version`功能标志。有关功能标志的更多信息，请参阅如何启用功能标记。

要从 CQv1 迁移到 CQv2，您必须创建新的队列策略或编辑将策略密钥定义设置为的现有队列`queue-version`策略。2有关应用策略的更多信息，请参阅策略。有关使用队列策略启用 CQv2 的更多信息，请参阅 RabbitMQ 文档中的 Classic Queues。

我们建议在开始迁移之前遵循其他最佳性能实践。

如果您使用的是队列策略，则删除队列策略会将 CQv2 队列降级回 CQv1。我们不建议将 CQv2 队列降级为 CQv1，因为 RabbitMQ 会转换队列的磁盘表示形式。对于较深的队列来说，这可能会占用大量内存，而且非常耗时。

Amazon MQ for RabbitMQ 中的限额

本主题列出了 Amazon MQ 中的配额。对于特定 AWS 账户，下面的许多配额可能会发生更改。要请求提高限制，请参阅《Amazon Web Services 一般参考》中的 AWS 服务限额。即使提高限制后，更新后的限制也将不可见。有关在 Amazon CloudWatch 中查看当前连接限制的更多信息，请参阅使用 Amazon CloudWatch 监控 Amazon MQ 代理。

主题

代理

下表列出了与 Amazon MQ for RabbitMQ 代理程序相关的限额。

限制	描述
代理名称	• 在代理区域和您的 AWS 账户中必须是唯一的。 • 长度必须介于 1 到 50 个字符之间。 • 只能包含 ASCII 可打印字符集中指定的字符。 • 只能包含字母数字字符、短划线、句点、下划线和波浪线 (- . _ ~)。
每个区域的代理数	50
每个代理的安全组	5
在 CloudWatch 中监控的 ActiveMQ 目标（队列和主题）	CloudWatch 只监控前 1000 个目标。
CloudWatch 中监控的 RabbitMQ 目标（队列）	CloudWatch 只监控前 500 个目标，按使用者数量排序。

限制	描述
每个代理的标签数	50

数据存储

下表列出了与 Amazon MQ for RabbitMQ 数据存储相关的限额。

限制	描述
每个小型代理的存储容量	`mq.*.micro` 实例类型代理为 20GB。有关 Amazon MQ 实例类型的更多信息，请参阅Broker instance types。
每个大型代理的存储容量	`mq.*.*large` 实例类型代理为 200GB。有关 Amazon MQ 实例类型的更多信息，请参阅Broker instance types。

API 限制

以下节流配额按AWS账户跨所有 Amazon MQ API 聚合，以维护服务带宽。要了解有关 Amazon MQ API 的更多信息，请参阅 Amazon MQ REST API 参考。

> **Important**
>
> 这些配额不适用于 Amazon MQ for ActiveMQ 或 Amazon MQ for RabbitMQ 代理消息 API。例如，Amazon MQ 不会限制消息的发送或接收。

API 突增限制	API 速率限制
100	15

Amazon MQ 中的安全性

AWS 的云安全性的优先级最高。为了满足对安全性最敏感的组织的需求，我们打造了具有超高安全性的数据中心和网络架构。作为 AWS 客户，您也将从这些数据中心和网络架构受益。

安全性是 AWS 和您的共同责任。责任共担模型将其描述为云的安全性和云中的安全性：

- 云的安全性 – AWS 负责保护在 AWS 云中运行 AWS 服务的基础设施。AWS 还向您提供可安全使用的服务。作为AWS 合规性计划的一部分，第三方审计人员将定期测试和验证安全性的有效性。要了解有关适用于 Amazon MQ 的合规性计划，请参阅合规性计划范围内的AWS服务。
- 云中的安全性 - 您的责任由您使用的 AWS 服务决定。您还需要对其他因素负责，包括您的数据的敏感性、您公司的要求以及适用的法律法规。

该文档帮助您了解如何在使用 Amazon MQ 时应用责任共担模式。以下主题说明如何配置 Amazon MQ 以实现您的安全性和合规性目标。您还会了解如何使用其他AWS服务来帮助您监控和保护 Amazon MQ 资源。

主题

Amazon MQ 中的数据保护

AWS责任共担模式适用于 Amazon MQ 中的数据保护。如该模式中所述，AWS 负责保护运行所有 AWS Cloud 的全球基础设施。您负责维护对托管在此基础设施上的内容的控制。您还负责您所使用的 AWS 服务 的安全配置和管理任务。有关数据隐私的更多信息，请参阅数据隐私常见问题。有关欧洲数据保护的信息，请参阅 AWS 安全性博客 上的 AWS 责任共担模式和 GDPR 博客文章。

出于数据保护目的，建议您保护 AWS 账户 凭证并使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 设置单个用户。这样，每个用户只获得履行其工作职责所需的权限。我们还建议您通过以下方式保护数据：

- 对每个账户使用多重身份验证 (MFA)。
- 使用 SSL/TLS 与 AWS 资源进行通信。我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 使用 AWS CloudTrail 设置 API 和用户活动日记账记录。
- 使用 AWS 加密解决方案以及 AWS 服务 中的所有默认安全控制。
- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Amazon S3 中的敏感数据。
- 如果在通过命令行界面或 API 访问 AWS 时需要经过 FIPS 140-2 验证的加密模块，请使用 FIPS 端点。有关可用的 FIPS 端点的更多信息，请参阅《美国联邦信息处理标准 (FIPS) 第 140-2 版》。

我们强烈建议您切勿将机密信息或敏感信息（如您客户的电子邮件地址）放入标签或自由格式文本字段（如 Name（名称）字段）。这包括当您通过控制台、API、AWS CLI 或 AWS SDK 使用 Amazon MQ 或其他 AWS 服务时。您在用于名称的标签或自由格式文本字段中输入的任何数据都可能会用于计费或诊断日记账。如果您向外部服务器提供 URL，我们强烈建议您不要在 URL 中包含凭证信息来验证您对该服务器的请求。

对于 Amazon MQ for ActiveMQ 和 Amazon MQ for RabbitMQ 代理，在通过代理 Web 控制台或 Amazon MQ API 创建资源时，请勿使用任何个人身份信息 (PII) 或其他机密或敏感信息作为代理名称或用户名。其他AWS服务（包括 CloudWatch Logs）可以访问代理名称和用户名。代理用户名不适合用于私有或敏感数据。

加密

Amazon MQ 中存储的用户数据进行静态加密。Amazon MQ 静态加密通过使用存储在 AWS Key Management Service（KMS）中的加密密钥来对数据加密，提供了增强的安全性。此服务可以帮助减少在保护敏感数据时涉及的操作负担和复杂性。通过静态加密，您可以构建符合加密合规性和法规要求的安全敏感型应用程序。

Amazon MQ 代理之间的所有连接都使用传输层安全性（TLS）在传输过程中提供加密。

Amazon MQ 使用其管理和安全存储的加密密钥对静态和传输中的消息进行加密。有关更多信息，请参阅 AWS Encryption SDK 开发人员指南。

静态加密

Amazon MQ 集成了 AWS Key Management Service（KMS）以提供透明的服务器端加密。Amazon MQ 始终加密您的静态数据。

当您创建 Amazon MQ for ActiveMQ 代理或 Amazon MQ for RabbitMQ 代理时，您可以指定您希望 Amazon MQ 用于加密您的静态数据的 AWS KMS key。如果您不指定 KMS 密钥，Amazon MQ 会为您创建一个 AWS 拥有的 KMS 密钥并代表您使用它。Amazon MQ 目前支持对称 KMS 密钥。有关 KMS 密钥的更多信息，请参阅 AWS KMS keys。

创建代理时，您可以通过选择以下选项之一来配置 Amazon MQ 用于加密密钥的内容。

- Amazon MQ owned KMS key (default) [Amazon MQ 拥有的 KMS 密钥(原定设置)] – 密钥归 Amazon MQ 拥有和管理，且不在您的账户中。
- AWS managed KMS key（AWS 托管式 KMS 密钥）– AWS 托管式 KMS 密钥 (`aws/mq`) 是您账户中的一个 KMS 密钥，由 Amazon MQ 代表您创建、管理和使用。
- 选择现有的客户托管式 KMS 密钥 – 客户托管式 CMK 由您在 AWS Key Management Service (KMS) 中创建和管理。

> **Important**
>
> - 撤消授权的操作无法撤消。相反，如果您需要撤消访问权限，我们建议您删除代理。
> - 对于使用 Amazon Elastic File System（EFS）存储消息数据的 Amazon MQ for ActiveMQ 代理，如果您撤消授予 Amazon EFS 在您的账户中使用 KMS 密钥的权限，则不会立即生效。
> - 对于使用 EBS 存储消息数据的 Amazon MQ for RabbitMQ 和 Amazon MQ for ActiveMQ 代理，如果您禁用、计划删除或撤消授予 Amazon EBS 在您的账户中使用 KMS 密钥的权限，Amazon MQ 将无法维护您的代理，且可能更改为降级状态。
> - 如果您已停用密钥或计划要删除的密钥，则可以重新激活密钥或取消密钥删除并维护您的代理。
> - 禁用密钥或撤消授权将不会立即发生。

使用适用于 RabbitMQ 的 KMS 密钥创建单实例代理程序时，您将看到 AWS CloudTrail 中记录了两个 `CreateGrant` 事件。第一个事件是 Amazon MQ 为 KMS 密钥创建授权。第二个事件是 EBS 创建授权供 EBS 使用。

CreateGrant AWS CloudTrail 日志条目：单实例代理程序

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

您将看到一个创建 EBS 授权的事件。

```
                                    {
    "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"
}
```

使用适用于 RabbitMQ 的 KMS 密钥创建集群部署时，您将看到 AWS CloudTrail 中记录了五个 `CreateGrant` 事件。前两个事件是为 Amazon MQ 创建授权。接下来的三个事件是 EBS 创建的供 EBS 使用的授权。

CreateGrant AWS CloudTrail 日志条目：集群部署

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

您将看到有关创建 EBS 授权的三个事件。

```
                                    {
    "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"
}
```

有关 KMS 密钥的更多信息，请参阅《AWS Key Management Service 开发人员指南》中的 AWS KMS keys。

传输中加密

Amazon MQ for ActiveMQ：Amazon MQ for ActiveMQ 需要强大的传输层安全性协议（TLS），并对 Amazon MQ 部署的代理之间的传输中数据进行加密。在 Amazon MQ 代理之间传递的所有数据均使用强大的传输层安全性协议（TLS）进行加密。这适用于所有可用的协议。

Amazon MQ for RabbitMQ：Amazon MQ for RabbitMQ 要求对所有客户端连接使用强大的传输层安全性协议（TLS）。RabbitMQ 集群复制流量仅传输代理的 VPC，AWS 数据中心之间的所有网络流量都

在物理层透明加密。Amazon MQ for RabbitMQ 集群代理目前不支持集群复制的节点间加密。要了解有关传输中数据的更多信息，请参阅加密静态数据和传输中数据。

Amazon MQ for ActiveMQ 协议

您可以使用以下启用 TLS 的协议来访问 ActiveMQ 代理：

- AMQP
- MQTT
- 基于 WebSocket 的 MQTT
- OpenWire
- STOMP
- 基于 WebSocket 的 STOMP

ActiveMQ 支持的 TLS 密码套件

Amazon MQ 上的 ActiveMQ 支持以下密码套件：

- 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 协议

您可以使用以下启用 TLS 的协议访问您的 RabbitMQ 代理：

- AMQP (0-9-1)

RabbitMQ 支持的 TLS 密码套件

Amazon MQ 上的 RabbitMQ 支持以下密码套件：

- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

适用于 Amazon MQ 的 Identity and Access Management

AWS Identity and Access Management (IAM) 是一项 AWS 服务，可以帮助管理员安全地控制对 AWS 资源的访问。IAM 管理员控制谁可以通过身份验证（登录）和授权（具有权限）使用 Amazon MQ 资源。IAM 是一项无需额外费用即可使用的 AWS 服务。

主题

- 受众
- 使用身份进行身份验证
- 使用策略管理访问
- Amazon MQ 如何与 IAM 协同工作
- Amazon MQ 基于身份的策略示例
- Amazon MQ 的 API 身份验证和授权
- AWS 亚马逊 MQ 的托管政策
- 对 Amazon MQ 使用服务相关角色

- Amazon MQ 身份和访问问题排查

受众

如何使用 AWS Identity and Access Management（IAM）因您在 Amazon MQ 中执行的操作而异。

服务用户 - 如果您使用 Amazon MQ 服务来完成工作，您的管理员会为您提供所需的凭证和权限。随着您使用更多 Amazon MQ 功能来完成工作，您可能需要额外权限。了解如何管理访问权限可帮助您向管理员请求适合的权限。如果您无法访问 Amazon MQ 中的功能，请参阅Amazon MQ 身份和访问问题排查。

服务管理员 – 如果您在公司负责管理 Amazon MQ 资源，您可能对 Amazon MQ 具有完全访问权限。您有责任确定您的服务用户应访问哪些 Amazon MQ 功能和资源。然后，您必须向 IAM 管理员提交请求以更改服务用户的权限。请查看该页面上的信息以了解 IAM 的基本概念。要了解有关您的公司如何将 IAM 与 Amazon MQ 搭配使用的更多信息，请参阅Amazon MQ 如何与 IAM 协同工作。

IAM 管理员 – 如果您是 IAM 管理员，您可能希望了解如何编写策略以管理对 Amazon MQ 的访问的详细信息。要查看您可在 IAM 中使用的 Amazon MQ 基于身份的策略示例，请参阅Amazon MQ 基于身份的策略示例。

使用身份进行身份验证

身份验证是您使用身份凭证登录 AWS 的方法。您必须作为 AWS 账户根用户、IAM 用户或通过担任 IAM 角色进行身份验证（登录到 AWS）。

您可以使用通过身份源提供的凭证以联合身份登录到 AWS。AWS IAM Identity Center(IAM Identity Center) 用户、您公司的单点登录身份验证以及您的 Google 或 Facebook 凭证都是联合身份的示例。当您以联合身份登录时，您的管理员以前使用 IAM 角色设置了身份联合验证。当您使用联合身份验证访问 AWS 时，您就是在间接担任角色。

根据您的用户类型，您可以登录 AWS Management Console 或 AWS 访问门户。有关登录到 AWS 的更多信息，请参阅《AWS 登录 用户指南》中的 如何登录到您的 AWS 账户。

如果您以编程方式访问 AWS，则 AWS 将提供软件开发工具包（SDK）和命令行界面（CLI），以便使用您的凭证以加密方式签署您的请求。如果您不使用 AWS 工具，则必须自行对请求签名。有关使用推荐的方法自行签署请求的更多信息，请参阅《IAM 用户指南》中的签署 AWS API 请求。

无论使用何种身份验证方法，您可能需要提供其他安全信息。例如，AWS 建议您使用多重身份验证 (MFA) 来提高账户的安全性。要了解更多信息，请参阅《AWS IAM Identity Center 用户指南》中的 多重身份验证 和《IAM 用户指南》中的 在 AWS 中使用多重身份验证（MFA）。

AWS 账户 根用户

当您创建 AWS 账户时，最初使用的是一个对账户中所有 AWS 服务和资源拥有完全访问权限的登录身份。此身份称为 AWS 账户根用户，使用您创建账户时所用的电子邮件地址和密码登录，即可获得该身份。强烈建议您不要使用根用户执行日常任务。保护好根用户凭证，并使用这些凭证来执行仅根用户可以执行的任务。有关要求您以根用户身份登录的任务的完整列表，请参阅《IAM 用户指南》中的需要根用户凭证的任务。

用户和组

IAM 用户是AWS 账户内对某个人员或应用程序具有特定权限的一个身份。在可能的情况下，我们建议使用临时凭证，而不是创建具有长期凭证（如密码和访问密钥）的 IAM 用户。但是，如果您有一些特定的使用场景需要长期凭证以及 IAM 用户，我们建议您轮换访问密钥。有关更多信息，请参阅《IAM 用户指南》中的 对于需要长期凭证的使用场景定期轮换访问密钥。

IAM 组是一个指定一组 IAM 用户的身份。您不能使用组的身份登录。您可以使用组来一次性为多个用户指定权限。如果有大量用户，使用组可以更轻松地管理用户权限。例如，您可能具有一个名为 IAMAdmins 的组，并为该组授予权限以管理 IAM 资源。

用户与角色不同。用户唯一地与某个人员或应用程序关联，而角色旨在让需要它的任何人担任。用户具有永久的长期凭证，而角色提供临时凭证。要了解更多信息，请参阅 IAM 用户指南中的何时创建 IAM 用户（而不是角色）。

IAM 角色

IAM 角色是AWS 账户中具有特定权限的身份。它类似于 IAM 用户，但与特定人员不关联。您可以通过切换角色，在 AWS Management Console 中暂时担任 IAM 角色。您可以调用 AWS CLI 或 AWS API 操作或使用自定义 URL 以担任角色。有关使用角色的方法的更多信息，请参阅《IAM 用户指南》中的 使用 IAM 角色。

具有临时凭证的 IAM 角色在以下情况下很有用：

- Federated user access（联合用户访问）- 要向联合身份分配权限，请创建角色并为角色定义权限。当联合身份进行身份验证时，该身份将与角色相关联并被授予由此角色定义的权限。有关联合身份验证的角色的信息，请参阅《IAM 用户指南》中的针对第三方身份提供程序创建角色。如果您使用 IAM Identity Center，则需要配置权限集。为控制您的身份在进行身份验证后可以访问的内容，IAM Identity Center 将权限集与 IAM 中的角色相关联。有关权限集的信息，请参阅《AWS IAM Identity Center 用户指南》中的 权限集。
- 临时 IAM 用户权限 - IAM 用户可担任 IAM 用户或角色，以暂时获得针对特定任务的不同权限。

- 跨账户访问 - 您可以使用 IAM 角色以允许不同账户中的某个人（可信主体）访问您的账户中的资源。角色是授予跨账户访问权限的主要方式。但是，对于某些 AWS 服务，您可以将策略直接附加到资源（而不是使用角色作为代理）。要了解用于跨账户访问的角色和基于资源的策略之间的差别，请参阅《IAM 用户指南》中的 IAM 角色与基于资源的策略有何不同。
- 跨服务访问 - 某些 AWS 服务使用其他 AWS 服务中的特征。例如，当您在某个服务中进行调用时，该服务通常会在 Amazon EC2 中运行应用程序或在 Amazon S3 中存储对象。服务可能会使用发出调用的主体的权限、使用服务角色或使用服务相关角色来执行此操作。
  - 转发访问会话：当您使用 IAM 用户或角色在 AWS 中执行操作时，您将被视为主体。使用某些服务时，您可能会执行一个操作，然后此操作在其他服务中启动另一个操作。FAS 使用主体调用 AWS 服务 的权限，结合请求的 AWS 服务，向下游服务发出请求。只有在服务收到需要与其他 AWS 服务 或资源交互才能完成的请求时，才会发出 FAS 请求。在这种情况下，您必须具有执行这两个操作的权限。有关发出 FAS 请求时的政策详情，请参阅转发访问会话。
  - 服务角色 - 服务角色是服务代表您在您的账户中执行操作而担任的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的 创建向 AWS 服务 委派权限的角色。
  - 服务相关角色 - 服务相关角色是与 AWS 服务关联的一种服务角色。服务可以担任代表您执行操作的角色。服务相关角色显示在您的 AWS 账户 中，并由该服务拥有。IAM 管理员可以查看但不能编辑服务相关角色的权限。
- 在 Amazon EC2 上运行的应用程序 - 您可以使用 IAM 角色管理在 EC2 实例上运行并发出 AWS CLI 或 AWS API 请求的应用程序的临时凭证。这优先于在 EC2 实例中存储访问密钥。要将 AWS 角色分配给 EC2 实例并使其对该实例的所有应用程序可用，您可以创建一个附加到实例的实例配置文件。实例配置文件包含角色，并使 EC2 实例上运行的程序能够获得临时凭证。有关更多信息，请参阅《IAM 用户指南》中的 使用 IAM 角色为 Amazon EC2 实例上运行的应用程序授予权限。

要了解是使用 IAM 角色还是 IAM 用户，请参阅 IAM 用户指南中的何时创建 IAM 角色（而不是用户）。

使用策略管理访问

您将创建策略并将其附加到 AWS 身份或资源，以控制 AWS 中的访问。策略是 AWS 中的对象；在与身份或资源相关联时，策略定义它们的权限。在主体（用户、根用户或角色会话）发出请求时，AWS 将评估这些策略。策略中的权限确定是允许还是拒绝请求。大多数策略在 AWS 中存储为 JSON 文档。有关 JSON 策略文档的结构和内容的更多信息，请参阅 IAM 用户指南 中的 JSON 策略概览。

管理员可以使用 AWS JSON 策略来指定谁有权访问什么内容。也就是说，哪个主体 可以对什么资源执行操作，以及在什么条件 下执行。

默认情况下，用户和角色没有权限。要授予用户对所需资源执行操作的权限，IAM 管理员可以创建 IAM policy。然后，管理员可以向角色添加 IAM policy，并且用户可以代入角色。

IAM 策略定义操作的权限，无关乎您使用哪种方法执行操作。例如，假设您有一个允许 `iam:GetRole` 操作的策略。具有该策略的用户可以从 AWS Management Console、AWS CLI 或 AWS API 获取角色信息。

基于身份的策略

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的 创建 IAM 策略。

基于身份的策略可以进一步归类为内联策略 或托管策略。内联策略直接嵌入单个用户、组或角色中。托管策略是可以附加到 AWS 账户 中的多个用户、组和角色的独立策略。托管策略包括 AWS 托管策略和客户托管策略。要了解如何在托管策略和内联策略之间进行选择，请参阅《IAM 用户指南》中的在托管策略与内联策略之间进行选择。

基于资源的策略

基于资源的策略是附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Amazon S3 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定主体。主体可以包括账户、用户、角色、联合身份用户或 AWS 服务。

基于资源的策略是位于该服务中的内联策略。您不能在基于资源的策略中使用来自 IAM 的 AWS 托管式策略。

访问控制列表（ACL）

访问控制列表 (ACL) 控制哪些主体（账户成员、用户或角色）有权访问资源。ACL 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

Amazon S3、AWS WAF 和 Amazon VPC 是支持 ACL 的服务示例。要了解有关 ACL 的更多信息，请参阅 Amazon Simple Storage Service 开发人员指南 中的访问控制列表 (ACL) 概览。

其他策略类型

AWS 支持额外的、不太常用的策略类型。这些策略类型可以设置更常用的策略类型向您授予的最大权限。

- 权限边界 - 权限边界是一个高级特征，用于设置基于身份的策略可以为 IAM 实体（IAM 用户或角色）授予的最大权限。您可为实体设置权限边界。这些结果权限是实体基于身份的策略及其权限边界的交集。在 `Principal` 中指定用户或角色的基于资源的策略不受权限边界限制。任一项策略中的显式拒绝将覆盖允许。有关权限边界的更多信息，请参阅 IAM 用户指南中的 IAM 实体的权限边界。
- 服务控制策略 (SCP) - SCP 是 JSON 策略，指定了组织或组织单位 (OU) 在 AWS Organizations 中的最大权限。AWS Organizations 服务可以分组和集中管理您的企业拥有的多个 AWS 账户 账户。如果在组织内启用了所有特征，则可对任意或全部账户应用服务控制策略 (SCP)。SCP 限制成员账户中实体（包括每个 AWS 账户根用户）的权限。有关 Organizations 和 SCP 的更多信息，请参阅 AWS Organizations 用户指南中的 SCP 的工作原理。
- 会话策略 - 会话策略是当您以编程方式为角色或联合身份用户创建临时会话时作为参数传递的高级策略。结果会话的权限是用户或角色的基于身份的策略和会话策略的交集。权限也可以来自基于资源的策略。任一项策略中的显式拒绝将覆盖允许。有关更多信息，请参阅 IAM 用户指南中的会话策略。

多个策略类型

当多个类型的策略应用于一个请求时，生成的权限更加复杂和难以理解。要了解 AWS 如何确定在涉及多种策略类型时是否允许请求，请参阅 IAM 用户指南中的策略评测逻辑。

Amazon MQ 如何与 IAM 协同工作

在使用 IAM 管理对 Amazon MQ 的访问权限之前，您应该了解哪些 IAM 功能可用于 Amazon MQ。要大致了解 Amazon MQ 和其他AWS服务如何与 IAM 一起使用，请参阅《IAM 用户指南》中的与 IAM 一起使用的AWS服务。

Amazon MQ 使用 IAM 来验证创建、更新和删除操作，但对代理进行本机 ActiveMQ 身份验证。有关更多信息，请参阅将 ActiveMQ 代理与 LDAP 集成。

主题

- Amazon MQ 基于身份的策略
- Amazon MQ 基于资源的策略
- 基于 Amazon MQ 标签的授权
- Amazon MQ IAM 角色

Amazon MQ 基于身份的策略

通过使用 IAM 基于身份的策略，您可以指定允许或拒绝的操作和资源以及允许或拒绝操作的条件。Amazon MQ 支持特定的操作、资源和条件键。要了解在 JSON 策略中使用的所有元素，请参阅 IAM 用户指南 中的 IAM JSON 策略元素参考。

操作

管理员可以使用 AWS JSON 策略来指定谁有权访问什么内容。也就是说，哪个主体 可以对什么资源执行操作，以及在什么 条件 下执行。

JSON 策略的 `Action` 元素描述可用于在策略中允许或拒绝访问的操作。策略操作通常与关联的 AWS API 操作同名。有一些例外情况，例如没有匹配 API 操作的仅限权限 操作。还有一些操作需要在策略中执行多个操作。这些附加操作称为相关操作。

在策略中包含操作以授予执行相关操作的权限。

Amazon MQ 中的策略操作在操作前面使用以下前缀：`mq:`。例如，要授予某人使用 Amazon MQ `CreateBroker` API 操作运行 Amazon MQ 实例的权限，您应将 `mq:CreateBroker` 操作纳入其策略。策略语句必须包含 `Action` 或 `NotAction` 元素。Amazon MQ 定义了一组自己的操作，以描述您可以使用该服务执行的任务。

要在单个语句中指定多项操作，请使用逗号将它们隔开，如下所示：

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

您也可以使用通配符 (*) 指定多个操作。例如，要指定以单词 `Describe` 开头的所有操作，包括以下操作：

```
"Action": "mq:Describe*"
```

要查看 Amazon MQ 操作的列表，请参阅《IAM 用户指南》中的 Amazon MQ 定义的操作。

资源

管理员可以使用 AWS JSON 策略来指定谁有权访问什么内容。也就是说，哪个主体 可以对什么资源执行操作，以及在什么条件 下执行。

Resource JSON 策略元素指定要向其应用操作的一个或多个对象。语句必须包含 Resource 或 NotResource 元素。作为最佳实践，请使用其 Amazon 资源名称（ARN）指定资源。对于支持特定资源类型（称为资源级权限）的操作，您可以执行此操作。

对于不支持资源级权限的操作（如列出操作），请使用通配符（*）指示语句应用于所有资源。

```
"Resource": "*"
```

在 Amazon MQ 中，主要AWS资源是 Amazon MQ 消息代理及其配置。每个 Amazon MQ 代理和配置都有与之关联的唯一 Amazon Resource Name（ARN），如下表所示。

资源类型	ARN	条件键
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}

有关 ARN 格式的更多信息，请参阅 Amazon Resource Name（ARN）和 AWS 服务命名空间。

例如，要在语句中指定名为 MyBroker 且 brokerId 为 b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9 的代理，请使用以下 ARN：

```
"Resource": "arn:aws:mq:us-
east-1:123456789012:broker:MyBroker:b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9"
```

要指定属于特定账户的所有代理和配置，请使用通配符（*）：

```
"Resource": "arn:aws:mq:us-east-1:123456789012:*"
```

无法对特定资源执行某些 Amazon MQ 操作，例如，用于创建资源的操作。在这些情况下，您必须使用通配符 (*)。

```
"Resource": "*"
```

API 操作 CreateTags 同时需要代理和配置。要在单个语句中指定多个资源，请使用逗号分隔 ARN。

```
"Resource": [
      "resource1",
      "resource2"
```

要查看 Amazon MQ 资源类型及其 ARN 的列表，请参阅《IAM 用户指南》中的 Amazon MQ 定义的资源。要了解您可以使用哪些操作指定每个资源的 ARN，请参阅 Amazon MQ 定义的操作。

条件键

管理员可以使用 AWS JSON 策略来指定谁有权访问什么内容。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

在 Condition 元素（或 Condition 块）中，可以指定语句生效的条件。Condition 元素是可选的。您可以创建使用条件运算符（例如，等于或小于）的条件表达式，以使策略中的条件与请求中的值相匹配。

如果您在一个语句中指定多个 Condition 元素，或在单个 Condition 元素中指定多个键，则 AWS 使用逻辑 AND 运算评估它们。如果您为单个条件键指定多个值，则 AWS 使用逻辑 OR 运算来评估条件。在授予语句的权限之前必须满足所有的条件。

在指定条件时，您也可以使用占位符变量。例如，只有在使用 IAM 用户名标记 IAM 用户时，您才能为其授予访问资源的权限。有关更多信息，请参阅《IAM 用户指南》中的 IAM policy 元素：变量和标签。

AWS支持全局条件键和特定于服务的条件键。要查看所有 AWS 全局条件键，请参阅《IAM 用户指南》中的 AWS 全局条件上下文键。

Amazon MQ 不定义任何特定于服务的条件键，但支持使用某些全局条件键。要查看 Amazon MQ 条件键的列表，请参阅下表，或参阅《IAM 用户指南》中的 Amazon MQ 的条件键。要了解您可以对哪些操作和资源使用条件键，请参阅 Amazon MQ 定义的操作。

条件键	描述	类型
aws:RequestTag/${TagKey}}	根据在请求中传递的标签筛选操作。	字符串
aws:ResourceTag/${TagKey}	根据与资源关联的标签筛选操作。	字符串

条件键	描述	类型
aws:TagKeys	根据在请求中传递的标签键筛选操作。	字符串

示例

要查看 Amazon MQ 基于身份的策略的示例，请参阅Amazon MQ 基于身份的策略示例。

Amazon MQ 基于资源的策略

目前，Amazon MQ 不支持使用基于资源的权限或基于资源的策略执行 IAM 身份验证。

基于 Amazon MQ 标签的授权

您可以将标签附加到 Amazon MQ 资源，或者在请求中将标签传递给 Amazon MQ。要基于标签控制访问，您需要使用 `mq:ResourceTag/key-name`、`aws:RequestTag/key-name` 或 `aws:TagKeys` 条件键在策略的条件元素中提供标签信息。

Amazon MQ 支持基于标签的策略。例如，您可能会拒绝对包含具有键 `environment` 和值 `production` 的标签的 Amazon MQ 资源的访问：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "mq:DeleteBroker",
                "mq:RebootBroker",
                "mq:DeleteTags"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/environment": "production"
                }
            }
        }
    ]
}
```

此策略将拒绝（Deny）删除或重启包含标签 environment/production 的 Amazon MQ 代理的能力。

有关标记的更多信息，请参阅：

- 为 资源添加标签
- 使用 IAM 标签控制访问

Amazon MQ IAM 角色

IAM 角色是 AWS 账户中具有特定权限的实体。

将临时凭证用于 Amazon MQ

您可以使用临时凭证进行联合身份登录，担任 IAM 角色或担任跨账户角色。您可以通过调用 AWS STS API 操作（如 AssumeRole 或 GetFederationToken）获得临时安全凭证。

Amazon MQ 支持使用临时凭证。

服务角色

此功能允许服务代表您担任服务角色。此角色允许服务访问其它服务中的资源以代表您完成操作。服务角色显示在您的 IAM 账户中，并归该账户所有。这意味着，IAM 管理员可以更改该角色的权限。但是，这样做可能会中断服务的功能。

Amazon MQ 支持服务角色。

Amazon MQ 基于身份的策略示例

原定设置情况下，用户和角色没有创建或修改 Amazon MQ 资源的权限。它们还无法使用 AWS Management Console、AWS CLI 或 AWS API 执行任务。IAM 管理员必须创建 IAM policy，以便为用户和角色授予权限以对所需的指定资源执行特定的 API 操作。然后，管理员必须将这些策略附加到需要这些权限的 IAM 用户或组。

要了解如何使用这些示例 JSON 策略文档创建 IAM 基于身份的策略，请参阅《IAM 用户指南》中的 在 JSON 选项卡上创建策略。

主题

- 策略最佳实践
- 使用 Amazon MQ 控制台

- 允许用户查看他们自己的权限

策略最佳实践

基于身份的策略确定某个人是否可以创建、访问或删除您账户中的 Amazon MQ 资源。这些操作可能会使 AWS 账户 产生成本。创建或编辑基于身份的策略时，请遵循以下准则和建议：

- AWS 托管策略及转向最低权限许可入门 - 要开始向用户和工作负载授予权限，请使用 AWS 托管策略来为许多常见使用场景授予权限。您可以在 AWS 账户 中找到这些策略。我们建议通过定义特定于您的使用场景的 AWS 客户管理型策略来进一步减少权限。有关更多信息，请参阅《IAM 用户指南》中的 AWS 托管策略或工作职能的 AWS 托管策略。
- 应用最低权限 – 在使用 IAM policy 设置权限时，请仅授予执行任务所需的权限。为此，您可以定义在特定条件下可以对特定资源执行的操作，也称为最低权限许可。有关使用 IAM 应用权限的更多信息，请参阅《IAM 用户指南》中的 IAM 中的策略和权限。
- 使用 IAM policy 中的条件进一步限制访问权限 – 您可以向策略添加条件来限制对操作和资源的访问。例如，您可以编写策略条件来指定必须使用 SSL 发送所有请求。如果通过特定 AWS 服务（例如 AWS CloudFormation）使用服务操作，您还可以使用条件来授予对服务操作的访问权限。有关更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略元素：条件。
- 使用 IAM Access Analyzer 验证您的 IAM policy，以确保权限的安全性和功能性 – IAM Access Analyzer 会验证新策略和现有策略，以确保策略符合 IAM policy语言 (JSON) 和 IAM 最佳实践。IAM Access Analyzer 提供 100 多项策略检查和可操作的建议，以帮助您制定安全且功能性强的策略。有关更多信息，请参阅《IAM 用户指南》中的 IAM Acess Analyzer 策略验证。
- Require multi-factor authentication（MFA）[需要多重身份验证（MFA）] – 如果您所处的场景要求您的 AWS 账户 中有 IAM 用户或根用户，请启用 MFA 来提高安全性。若要在调用 API 操作时需要 MFA，请将 MFA 条件添加到您的策略中。有关更多信息，请参阅《IAM 用户指南》中的配置受 MFA 保护的 API 访问。

有关 IAM 中的最佳实践的更多信息，请参阅《IAM 用户指南》中的 IAM 中的安全最佳实践。

使用 Amazon MQ 控制台

要访问 Amazon MQ 控制台，您必须具有一组最低的权限。这些权限必须允许您列出和查看有关您AWS账户中的 Amazon MQ 资源的详细信息。如果您创建的基于身份的策略比所需的最低权限更严格，则无法为具有该策略的实体（IAM 用户或角色）正常运行控制台。

要确保这些实体仍可使用 Amazon MQ 控制台，也可向实体附加以下AWS托管式策略。有关更多信息，请参阅《IAM 用户指南》中的为用户添加权限：

```
AmazonMQReadOnlyAccess
```

对于只需要调用 AWS CLI 或 AWS API 的用户，无需为其提供最低控制台权限。相反，只允许访问与您尝试执行的 API 操作相匹配的操作。

允许用户查看他们自己的权限

该示例说明了您如何创建策略，以允许 IAM 用户查看附加到其用户身份的内联和托管式策略。此策略包括在控制台上完成此操作或者以编程方式使用 AWS CLI 或 AWS API 所需的权限。

```
{
    "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 的 API 身份验证和授权

Amazon MQ 对 API 身份验证使用标准AWS请求签名。有关更多信息，请参阅 AWS 中的签署 AWS 一般参考 API 请求。

> **Note**
>
> 目前，Amazon MQ 不支持使用基于资源的权限或基于资源的策略执行 IAM 身份验证。

要授权 AWS 用户使用代理、配置和用户，必须编辑您的 IAM 策略权限。

主题

- 要创建 Amazon MQ 代理所需的 IAM 权限
- Amazon MQ REST API 权限参考
- Amazon MQ API 操作的资源级权限

要创建 Amazon MQ 代理所需的 IAM 权限

要创建代理，您必须使用 `AmazonMQFullAccess` IAM 策略或在 IAM 策略中包含以下 EC2 权限。

以下自定义策略包含两个语句（其中一个为条件语句），可授予用于操作 Amazon MQ 创建 ActiveMQ 代理所需的资源的权限。

> **Important**
>
> - 要允许 Amazon MQ 代表您在您的账户中创建弹性网络接口（ENI），`ec2:CreateNetworkInterface` 操作是必需的。
> - `ec2:CreateNetworkInterfacePermission` 操作授权 Amazon MQ 将 ENI 附加到 ActiveMQ 代理。
> - `ec2:AuthorizedService` 条件键确保 ENI 权限只能授予给 Amazon MQ 服务账户。

```
{
    "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"
            }
        }
    }]
}
```

有关更多信息，请参阅第 2 步：创建用户并获取您的 AWS 证书和永远不要修改或删除 Amazon MQ 弹性网络接口。

Amazon MQ REST API 权限参考

下表列出了 Amazon MQ REST API 以及相应的 IAM 权限。

Amazon MQ REST API 和必需权限

Amazon MQ REST API	所需权限
CreateBroker	mq:CreateBroker
CreateConfiguration	mq:CreateConfiguration

Amazon MQ REST API	所需权限
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
UpdateUser	mq:UpdateUser

Amazon MQ API 操作的资源级权限

术语资源级权限指的是能够指定允许用户对哪些资源执行操作的能力。Amazon MQ 部分支持资源级权限。对于某些 Amazon MQ 操作，您可以控制何时允许用户执行操作（基于必须满足的条件）或是允许用户使用的特定资源。

下表介绍当前支持资源级权限的 Amazon MQ API 操作，以及每个操作支持的资源、资源 ARN 和条件键。

> **Important**
>
> 如果某一 Amazon MQ API 操作未在此表中列出，则表示它不支持资源级权限。如果 Amazon MQ API 操作不支持资源级权限，则您可以向用户授予使用该操作的权限，但是必须为策略语句的资源元素指定 * 通配符。

API 操作	资源类型（* 为必需）
CreateConfiguration	配置*
CreateTags	代理、配置
CreateUser	代理*
DeleteBroker	代理*
DeleteUser	代理*
DescribeBroker	代理*
DescribeConfiguration	配置*
DescribeConfigurationRevision	配置*
DescribeUser	代理*
ListConfigurationRevisions	配置*
ListConfigurationRevisions	配置*
ListTags	代理、配置
ListUsers	代理*

API 操作	资源类型（* 为必需）
RebootBroker	代理*
UpdateBroker	代理*
UpdateConfiguration	配置*
UpdateUser	代理*

AWS 亚马逊 MQ 的托管政策

AWS 托管策略是由创建和管理的独立策略 AWS。 AWS 托管策略旨在为许多常见用例提供权限，以便您可以开始为用户、组和角色分配权限。

请记住， AWS 托管策略可能不会为您的特定用例授予最低权限权限，因为它们可供所有 AWS 客户使用。我们建议通过定义特定于您的使用场景的客户管理型策略来进一步减少权限。

您无法更改 AWS 托管策略中定义的权限。如果 AWS 更新 AWS 托管策略中定义的权限，则更新会影响该策略所关联的所有委托人身份（用户、组和角色）。 AWS 最有可能在启动新的 API 或现有服务可以使用新 AWS 服务 的 API 操作时更新 AWS 托管策略。

有关更多信息，请参阅《IAM 用户指南》中的 AWS 托管式策略。

AWS 托管策略：亚马逊 MQ ServiceRole 政策

您不能将 AmazonMQServiceRolePolicy 附加到您的 IAM 实体。将此策略附加到允许 Amazon MQ 代表您执行操作的服务相关角色。有关此权限策略及其允许 Amazon MQ 执行的操作的更多信息，请参阅the section called “Amazon MQ 的服务相关角色权限”。

亚马逊 MQ 更新了托管政策 AWS

查看自该服务开始跟踪这些更改以来，Amazon MQ AWS 托管政策更新的详细信息。有关此页面更改的自动提示，请订阅 Amazon MQ 文档历史记录页面上的 RSS 源。

更改	描述	日期
Amazon MQ 已开始跟踪更改	Amazon MQ 开始跟踪其 AWS 托管政策的变更。	2021 年 5 月 5 日

对 Amazon MQ 使用服务相关角色

Amazon MQ 使用 AWS Identity and Access Management（IAM）服务相关角色。服务相关角色是一种独特类型的 IAM 角色，它与 Amazon MQ 直接相关。服务相关角色是由 Amazon MQ 预定义的，并包含服务代表您调用其他AWS服务所需的所有权限。

服务相关角色可让您更轻松地设置 Amazon MQ，因为您不必手动添加必要的权限。Amazon MQ 定义其服务相关角色的权限，除非另外定义，否则只有 Amazon MQ 可以代入该角色。定义的权限包括信任策略和权限策略，以及不能附加到任何其它 IAM 实体的权限策略。

只有在首先删除相关资源后，才能删除服务相关角色。这将保护您的 Amazon MQ 资源，因为您不会无意中删除对资源的访问权限。

有关支持服务相关角色的其他服务的信息，请参阅与 IAM 配合使用的 AWS 服务，并在服务相关角色列中查找显示为 Yes（是）的服务。选择 Yes (是) 和链接，查看该服务的服务相关角色文档。

Amazon MQ 的服务相关角色权限

Amazon MQ 使用名为 AWSServiceRoleForAmazonMQ 的服务相关角色 – Amazon MQ 使用此服务相关角色代表您调用AWS服务。

AWSServiceRoleForAmazonMQ 服务相关角色信任以下服务以担任该角色：

- `mq.amazonaws.com`

Amazon MQ 使用附加到 AWSServiceRoleForAmazonMQ 服务相关角色的权限策略 `AmazonMQServiceRolePolicy` 来完成对指定资源的以下操作：

- 操作：对 `vpc` 资源执行的 `ec2:CreateVpcEndpoint`。
- 操作：对 `subnet` 资源执行的 `ec2:CreateVpcEndpoint`。
- 操作：对 `security-group` 资源执行的 `ec2:CreateVpcEndpoint`。

- 操作：对 vpc-endpoint 资源执行的 ec2:CreateVpcEndpoint。
- 操作：对 vpc 资源执行的 ec2:DescribeVpcEndpoints。
- 操作：对 subnet 资源执行的 ec2:DescribeVpcEndpoints。
- 操作：对 vpc-endpoint 资源执行的 ec2:CreateTags。
- 操作：对 log-group 资源执行的 logs:PutLogEvents。
- 操作：对 log-group 资源执行的 logs:DescribeLogStreams。
- 操作：对 log-group 资源执行的 logs:DescribeLogGroups。
- 操作：对 log-group 资源执行的 CreateLogStream。
- 操作：对 log-group 资源执行的 CreateLogGroup。

当您创建 Amazon MQ for RabbitMQ 代理时，AmazonMQServiceRolePolicy 权限策略允许 Amazon MQ 代表您执行以下任务。

- 使用您提供的 Amazon VPC、子网和安全组为代理创建 Amazon VPC 终端节点。您可以使用为代理创建的终端节点通过 RabbitMQ 管理控制台、管理 API 或以编程方式连接到代理。
- 创建日志组，并将代理日志发布到 Amazon CloudWatch Logs。

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

必须配置权限，允许 IAM 实体（如用户、组或角色）创建、编辑或删除服务相关角色。有关更多信息，请参阅《IAM 用户指南》中的服务相关角色权限。

为 Amazon MQ 创建服务相关角色

无需手动创建服务相关角色。当您首次创建代理时，Amazon MQ 将创建服务相关角色以调用AWS服务。您创建的所有后续代理都将使用相同的角色，并且不会创建新角色。

> **Important**
>
> 如果您在其他使用此角色支持的功能的服务中完成某个操作，此服务相关角色可以出现在您的账户中。要了解更多信息，请参阅我的 IAM 账户中出现新角色。

如果删除此服务相关角色，然后需要再次创建，可以使用相同流程在账户中重新创建此角色。

您也可以使用 IAM 控制台为 Amazon MQ 使用案例创建服务相关角色。在 AWS CLI 或 AWS API 中，使用 `mq.amazonaws.com` 服务名称创建服务相关角色。有关更多信息，请参阅《IAM 用户指南》 中的创建服务相关角色。如果您删除了此服务相关角色，则可以使用此相同过程再次创建角色。

为 Amazon MQ 编辑服务相关角色

Amazon MQ 不允许您编辑 AWSServiceRoleForAmazonMQ 服务相关角色。但是可以使用 IAM 编辑角色说明。有关更多信息，请参阅 IAM 用户指南中的编辑服务相关角色。

删除适用于 Amazon MQ 的服务相关角色

如果不再需要使用某个需要服务相关角色的功能或服务，我们建议您删除该角色。这样您就没有未被主动监控或维护的未使用实体。但是，您必须先清除服务相关角色的资源，然后才能手动删除它。

> **Note**
>
> 如果在您试图删除资源时，Amazon MQ 服务正在使用该角色，则删除操作可能会失败。如果发生这种情况，请等待几分钟后重试。

删除 AWSServiceRoleForAmazonMQ 使用的 Amazon MQ 资源

- 使用AWS Management Console、Amazon MQ CLI 或 Amazon MQ API 删除您的 Amazon MQ 代理。有关删除代理的更多信息，请参阅???。

使用 IAM 手动删除服务相关角色

使用 IAM 控制台、AWS CLI 或 AWS API 删除 AWSServiceRoleForAmazonMQ 服务相关角色。有关更多信息，请参见 IAM 用户指南中的删除服务相关角色。

Amazon MQ 服务相关角色支持的区域

Amazon MQ 支持在该服务可用的所有区域中使用服务相关角色。有关更多信息，请参阅 AWS 区域和终端节点。

区域名称	区域标识	Amazon MQ 支持
美国东部（弗吉尼亚北部）	us-east-1	是
美国东部（俄亥俄）	us-east-2	是
美国西部（加利福尼亚北部）	us-west-1	是
美国西部（俄勒冈）	us-west-2	是
Asia Pacific (Mumbai)	ap-south-1	是
亚太地区（大阪）	ap-northeast-3	是

区域名称	区域标识	Amazon MQ 支持
亚太地区（首尔）	ap-northeast-2	是
亚太地区（新加坡）	ap-southeast-1	是
亚太地区（悉尼）	ap-southeast-2	是
亚太地区（东京）	ap-northeast-1	是
Canada (Central)	ca-central-1	是
欧洲（法兰克福）	eu-central-1	是
欧洲（爱尔兰）	eu-west-1	是
欧洲（伦敦）	eu-west-2	是
欧洲（巴黎）	eu-west-3	是
South America (São Paulo)	sa-east-1	是
AWS GovCloud (US)	us-gov-west-1	否

Amazon MQ 身份和访问问题排查

可以使用以下信息，以帮助您诊断和修复在使用 Amazon MQ 和 IAM 时可能遇到的常见问题。

主题

- 我无权在 Amazon MQ 中执行操作
- 我无权执行 iam:PassRole
- 我希望允许我的AWS账户以外的人访问我的 Amazon MQ 资源

我无权在 Amazon MQ 中执行操作

如果 AWS Management Console 告诉您，您无权执行某个操作，则必须联系您的管理员寻求帮助。管理员是向您提供登录凭证的人。

当 mateojackson 用户尝试使用控制台查看有关 *widget* 的详细信息但不具有 mq:*GetWidget* 权限时，会发生以下示例错误。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 mq:GetWidget on resource: my-example-widget
```

在这种情况下，Mateo 请求他的管理员更新其策略，以允许他使用 mq:*GetWidget* 操作访问 *my-example-widget* 资源。

我无权执行 iam:PassRole

如果您收到一个错误，表明您无权执行 iam:PassRole 操作，则必须更新策略以允许您将角色传递给 Amazon MQ。

有些 AWS 服务允许您将现有角色传递到该服务，而不是创建新服务角色或服务相关角色。为此，您必须具有将角色传递到服务的权限。

当名为 marymajor 的 IAM 用户尝试使用控制台在 Amazon MQ 中执行操作时，会发生以下示例错误。但是，服务必须具有服务角色所授予的权限才可执行此操作。Mary 不具有将角色传递到服务的权限。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在这种情况下，必须更新 Mary 的策略以允许她执行 iam:PassRole 操作。

如果您需要帮助，请联系您的 AWS 管理员。管理员是向您提供登录凭证的人。

我希望允许我的AWS账户以外的人访问我的 Amazon MQ 资源

您可以创建一个角色，以便其它账户中的用户或您组织外的人员可以使用该角色来访问您的资源。您可以指定谁值得信赖，可以担任角色。对于支持基于资源的策略或访问控制列表 (ACL) 的服务，您可以使用这些策略向人员授予对您的资源的访问权。

要了解更多信息，请参阅以下内容：

- 要了解 Amazon MQ 是否支持这些功能，请参阅Amazon MQ 如何与 IAM 协同工作。
- 要了解如何为您拥有的 AWS 账户 中的资源提供访问权限，请参阅 IAM 用户指南中的为您拥有的另一个 AWS 账户 中的 IAM 用户提供访问权限。

- 要了解如何为第三方AWS 账户提供您的资源的访问权限，请参阅 IAM 用户指南中的为第三方拥有的 AWS 账户提供访问权限。
- 要了解如何通过联合身份验证提供访问权限，请参阅 IAM 用户指南中的为经过外部身份验证的用户（联合身份验证）提供访问权限。
- 要了解使用角色和基于资源的策略进行跨账户访问之间的差别，请参阅 IAM 用户指南中的 IAM 角色与基于资源的策略有何不同。

Amazon MQ 的合规性验证

作为多项合规计划的一部分，第三方审计师评估 Amazon MQ 的安全与 AWS 合规性。其中包括 SOC、PCI、HIPAA 等。

要了解是否属于特定合规计划的范围，请参阅AWS 服务 “按合规计划划分的范围” ”，然后选择您感兴趣的合规计划。 AWS 服务 有关一般信息，请参阅AWS 合规计划AWS。

您可以使用下载第三方审计报告 AWS Artifact。有关更多信息，请参阅中的 “下载报告” 中的 “ AWS Artifact。

您在使用 AWS 服务 时的合规责任取决于您的数据的敏感性、贵公司的合规目标以及适用的法律和法规。 AWS 提供了以下资源来帮助实现合规性：

- 安全与合规性快速入门指南 — 这些部署指南讨论了架构注意事项，并提供了部署以安全性和合规性为重点 AWS 的基准环境的步骤。
- 在 A@@ mazon Web Services 上构建 HIPAA 安全与合规性 — 本白皮书描述了各公司如何使用 AWS 来创建符合 HIPAA 资格的应用程序。

> **Note**
>
> 并非所有 AWS 服务 人都符合 HIPAA 资格。有关更多信息，请参阅符合 HIPAA 要求的服务参考。

- AWS 合AWS 规资源 — 此工作簿和指南集合可能适用于您的行业和所在地区。
- AWS 客户合规指南 — 从合规角度了解责任共担模式。这些指南总结了保护的最佳实践， AWS 服务并将指南映射到跨多个框架（包括美国国家标准与技术研究院 (NIST)、支付卡行业安全标准委员会 (PCI) 和国际标准化组织 (ISO)）的安全控制。
- 使用AWS Config 开发人员指南中的规则评估资源 — 该 AWS Config 服务评估您的资源配置在多大程度上符合内部实践、行业准则和法规。

- AWS Security Hub— 这 AWS 服务 可以全面了解您的安全状态 AWS。Security Hub 通过安全控件评估您的 AWS 资源并检查其是否符合安全行业标准和最佳实践。有关受支持服务及控件的列表，请参阅 Security Hub 控件参考。
- Amazon GuardDuty — 它通过监控您的 AWS 账户环境中是否存在可疑和恶意活动，来 AWS 服务检测您的工作负载、容器和数据面临的潜在威胁。 GuardDuty 通过满足某些合规性框架规定的入侵检测要求，可以帮助您满足各种合规性要求，例如 PCI DSS。
- AWS Audit Manager— 这 AWS 服务 可以帮助您持续审计 AWS 使用情况，从而简化风险管理以及对法规和行业标准的合规性。

Amazon MQ 中的恢复能力

AWS全球基础设施围绕AWS区域和可用区构建。AWS区域提供多个在物理上独立且隔离的可用区，这些可用区通过延迟低、吞吐量高且冗余性高的网络连接在一起。利用可用区，您可以设计和操作在可用区之间无中断地自动实现故障转移的应用程序和数据库。与传统的单个或多个数据中心基础设施相比，可用区具有更高的可用性、容错性和可扩展性。

有关 AWS 区域和可用区的更多信息，请参阅 AWS 全球基础设施。

Amazon MQ 中的基础设施安全性

作为一项托管式服务，受 AWS 全球网络安全保护。有关 AWS 安全服务以及 AWS 如何保护基础架构的信息，请参阅 AWS 云安全。要按照基础设施安全最佳实践设计您的 AWS 环境，请参阅《安全性支柱 AWS Well-Architected Framework》中的 基础设施保护。

您可以使用 AWS 发布的 API 调用通过网络进行访问。客户端必须支持以下内容：

- 传输层安全性协议（TLS） 我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 具有完全向前保密 (PFS) 的密码套件，例如 DHE（Ephemeral Diffie-Hellman）或 ECDHE（Elliptic Curve Ephemeral Diffie-Hellman）。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

此外，必须使用访问密钥 ID 和与 IAM 委托人关联的秘密访问密钥来对请求进行签名。或者，您可以使用 AWS Security Token Service (AWS STS) 生成临时安全凭证来对请求进行签名。

Amazon MQ 的安全最佳实践

以下设计模式可以提高 Amazon MQ 代理的安全性。

主题

有关 Amazon MQ 如何加密您数据的更多信息以及支持的协议列表，请参阅数据保护。

首选不可公开访问的代理

对于所创建的、不可公开访问的代理，无法从您的 VPC 外部访问。这可大大降低您的代理遭受来自公共 Internet 的分布式拒绝服务 (DDoS) 攻击的危险。有关更多信息，请参阅本指南中的访问不可公开访问的代理 Web 控制台和 安全博客中的如何通过减少攻击面来帮助应对 DDoS 攻击AWS。

始终配置授权映射

由于 ActiveMQ 默认情况下没有配置授权映射，任何经过身份验证的用户都可以在代理上执行任何操作。因此，最好按组 来限制权限。有关更多信息，请参阅`authorizationEntry`。

> **Important**
>
> 如果您指定的授权映射不包含在 `activemq-webconsole` 组中，您无法使用 ActiveMQ Web 控制台，因为该组未获得授权向 Amazon MQ 代理发送消息或接收来自该代理的消息。

使用 VPC 安全组阻止不必要的协议

为了提高安全性，您应通过正确配置 Amazon VPC 安全组来限制不必要的协议和端口的连接。例如，要在允许访问 OpenWire 和 Web 控制台的同时限制对大多数协议的访问，您可以仅允许访问 61617 和 8162。这会通过阻止您不使用的协议来限制您的暴露，同时允许 OpenWire 和 Web 控制台正常运行。

仅允许您正在使用的协议端口。

- AMQP: 5671
- MQTT: 8883
- OpenWire: 61617
- STOMP: 61614
- WebSocket: 61619

有关更多信息，请参阅：

- Configure Additional Broker Settings
- 您的 VPC 的安全组
- 您的 VPC 的默认安全组
- 使用安全组

监控和记录 Amazon MQ 代理

监控是保持AWS解决方案的可靠性、可用性和性能的重要方面。您应该从AWS解决方案的各个部分收集监控数据，以便您可以更轻松地调试多点故障（如果发生）。AWS提供多种工具来监控您的 Amazon MQ 资源并对潜在事件做出响应：

主题

访问 Amazon MQ 的 CloudWatch 指标

Amazon MQ 和 Amazon CloudWatch 集成在一起，因此您可以使用 CloudWatch 查看和分析 ActiveMQ 代理和代理的目标（队列和主题）的指标。您可以从 CloudWatch 控制台、AWS CLI 或 CloudWatch CLI 查看和分析 Amazon MQ 指标。每分钟自动从代理中轮询一次 Amazon MQ 的 CloudWatch 指标，然后将其推送到 CloudWatch。

有关 Amazon MQ 指标的完整列表，请参阅Monitoring Amazon MQ using CloudWatch。

有关为指标创建 CloudWatch 警报的信息，请参阅《Amazon CloudWatch 用户指南》中的创建或编辑 CloudWatch 警报。

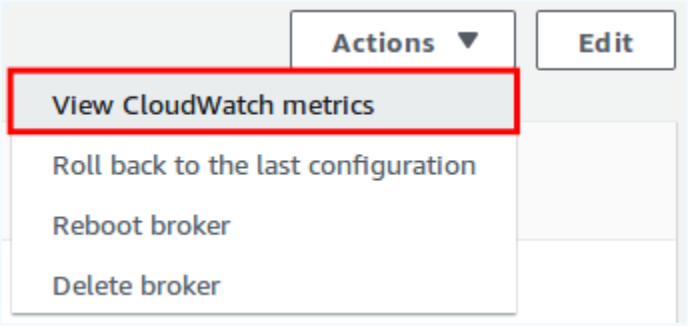
> **Note**
>
> 通过 CloudWatch 报告 Amazon MQ 指标无需任何费用。这些指标作为 Amazon MQ 服务的一部分提供。
> 对于 ActiveMQ 代理，CloudWatch 只监控前 1000 个目标。
> 对于 RabbitMQ 代理，CloudWatch 只监控前 500 个目标，按使用者数量排序。

主题

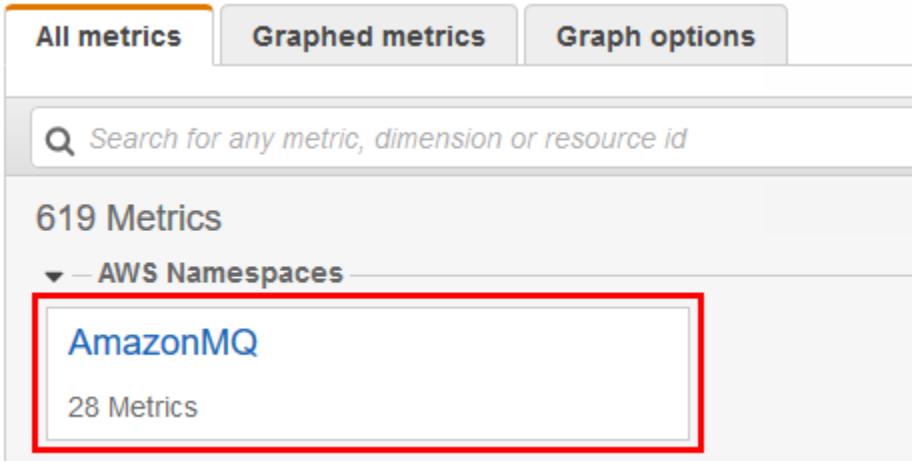
- Amazon CloudWatch API

AWS Management Console

以下示例显示如何使用AWS Management Console访问 Amazon MQ 的 CloudWatch 指标。

> **Note**
>
> 如果您已登录到 Amazon MQ 控制台，请在代理 Details (详细信息) 页面上，依次选择 Actions (操作)、View CloudWatch metrics (查看 CloudWatch 指标)。
>
> 
>

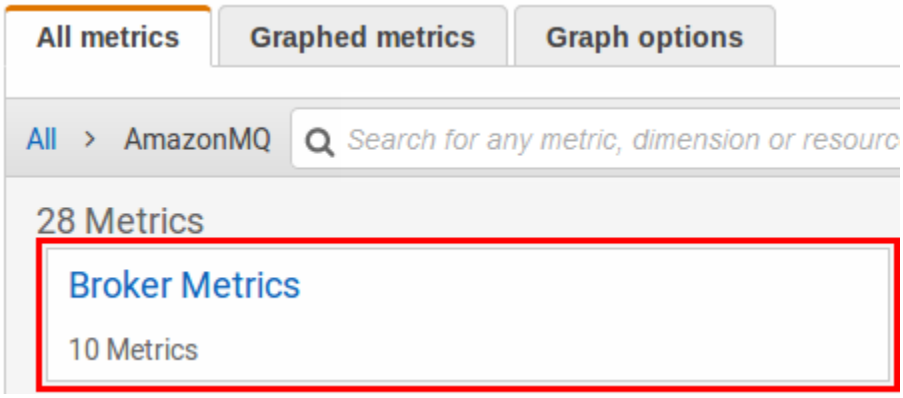
1. 登录到 CloudWatch 控制台。
2. 在导航面板上，选择 Metrics。
3. 选择 AmazonMQ 指标命名空间。



4. 选择以下指标维度之一：
   - 代理指标
   - 代理的队列指标

- 代理的主题指标

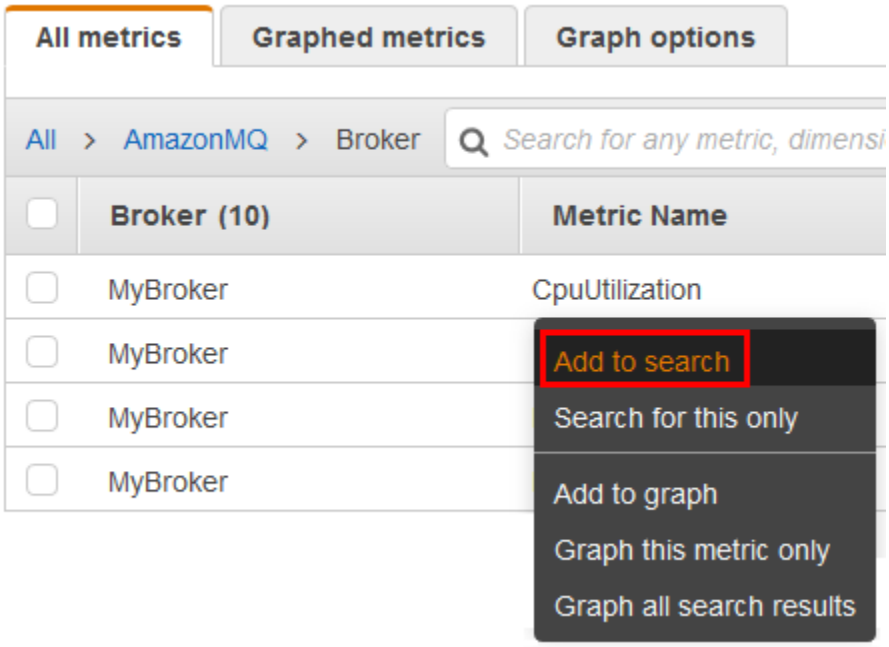
在此示例中，选择了 Broker Metrics (代理指标)。



5. 现在可检查 Amazon MQ 指标：

- 要对指标进行排序，请使用列标题。

- 要为指标绘制图表，请选中该指标旁边的复选框。
- 要按指标进行筛选，请选择指标名称，然后选择 Add to search。



AWS Command Line Interface

要使用 AWS CLI 访问 Amazon MQ 指标，请使用 `get-metric-statistics` 命令。

有关更多信息，请参阅《Amazon CloudWatch 用户指南》中的获取指标的统计信息。

Amazon CloudWatch API

要使用 CloudWatch API 访问 Amazon MQ 指标，请使用 `GetMetricStatistics` 操作。

有关更多信息，请参阅《Amazon CloudWatch 用户指南》中的获取指标的统计信息。

使用 Amazon CloudWatch 监控 Amazon MQ 代理

Amazon MQ 和 Amazon CloudWatch 集成在一起，因此您可以使用 CloudWatch 查看和分析 ActiveMQ 代理和代理的目标（队列和主题）的指标。您可以从 CloudWatch 控制台、AWS CLI 或 CloudWatch CLI 查看和分析 Amazon MQ 指标。每分钟自动从代理中轮询一次 Amazon MQ 的 CloudWatch 指标，然后将其推送到 CloudWatch。

有关信息，请参阅 访问 Amazon MQ 的 CloudWatch 指标。

> **Note**
>
> 以下统计数据对所有指标都有效：
>
> - `Average`
> - `Minimum`
> - `Maximum`
> - `Sum`

`AWS/AmazonMQ` 命名空间包括以下指标。

主题

- 记录和监控 Amazon MQ for ActiveMQ 代理
- 记录和监控 Amazon MQ for RabbitMQ 代理

记录和监控 Amazon MQ for ActiveMQ 代理

Amazon MQ for ActiveMQ 指标

指标	Unit	描述
`AmqpMaximumConnections`	计数	您可以使用 AMQP 连接到代理的最大客户端数量。有关连接配额的更多信息，请参阅 Quotas in Amazon MQ。
`BurstBalance`	百分比	Amazon EBS 卷上剩余的突增积分百分比，用于保留吞吐量优化代理的消息数据。如果此余额达到零，Amazon EBS 卷提供的 IOPS 将减少，直到突增余额重新填充。有关 Amazon EBS 中突增余额如何使用的更多信息，请参阅：I/O 积分和突增性能。
`CpuCreditBalance`	积分（vCPU 分钟）	**Important** 该指标仅适用于 `mq.t2.micro` 代理实例类型。 CPU 积分指标仅每 5 分钟提供一次。 实例自启动后已累积获得的 CPU 积分数（包括启动积分数）。积分余额可供代理实例用于支付超出基准 CPU 利用率的突增部分。 在获得积分后，积分将在积分余额中累积；在花费积分后，

指标	Unit	描述
		将从积分余额中扣除。积分余额有上限。达到该限制后，新获得的积分将被丢弃。
CpuUtilization	百分比	代理当前正在使用的已分配 Amazon EC2 计算单位的百分比。
CurrentConnectionsCount	计数	当前代理上的活动连接数量。
EstablishedConnectionsCount	计数	已在代理上建立的活动和非活动连接总数。
HeapUsage	百分比	代理当前使用的 ActiveMQ JVM 内存限制的百分比。
InactiveDurableTopicSubscribersCount	计数	非活动持久主题订阅者的数量，最多可达 2000。
JobSchedulerStorePercentUsage	百分比	作业调度程序存储所使用的磁盘空间的百分比。
JournalFilesForFastRecovery	计数	干净关闭后将重放的日志文件数。
JournalFilesForFullRecovery	计数	不干净关闭后将重放的日志文件数。
MqttMaximumConnections	计数	您可以使用 MQTT 连接到代理的最大客户端数量。有关连接配额的更多信息，请参阅 Quotas in Amazon MQ。
NetworkConnectorConnectionCount	计数	使用 NetworkConnector 连接到代理网络中的代理的节点数。

指标	Unit	描述
NetworkIn	字节	代理的传入流量。
NetworkOut	字节	代理的传出流量。
OpenTransactionCount	计数	正在进行的事务总数。
OpenwireMaximumConnections	计数	您可以使用 OpenWire 连接到代理的最大客户端数量。有关连接配额的更多信息，请参阅 Quotas in Amazon MQ。
StompMaximumConnections	计数	您可以使用 STOMP 连接到代理的最大客户端数量。有关连接配额的更多信息，请参阅 Quotas in Amazon MQ。
StorePercentUsage	百分比	存储限制使用的百分比。如果此数值达到 100，代理将拒绝消息。
TempPercentUsage	百分比	非持久性消息使用的可用临时存储的百分比。
TotalConsumerCount	计数	订阅当前代理目标的消息使用者数量。
TotalMessageCount	计数	存储在代理上的消息数量。
TotalProducerCount	计数	在当前代理上的目标上处于活动状态的消息创建器数量。
VolumeReadOps	计数	在 Amazon EBS 卷上进行的读取操作数。
VolumeWriteOps	计数	在 Amazon EBS 卷上进行的写入操作数。

指标	Unit	描述
WsMaximumConnections	计数	您可以使用 WebSocket 连接到代理的最大客户端数量。有关连接配额的更多信息，请参阅 Quotas in Amazon MQ。

ActiveMQ 代理指标的维度

维度	描述
Broker	代理的名称 **Note** 单实例代理具有后缀 -1。高可用性的主动/备用代理具有后缀 -1 和 -2 可用于其冗余对。

ActiveMQ 目标（队列和主题）指标

Important

以下指标包括 CloudWatch 轮询周期的每分钟计数。

- EnqueueCount
- ExpiredCount
- DequeueCount
- DispatchCount
- InFlightCount

例如，在五分钟的 CloudWatch 期间，EnqueueCount 有五个计数值，每分钟对应一个值。Minimum 和 Maximum 统计数据提供指定期间内的最低和最高每分钟值。

指标	Unit	描述
ConsumerCount	计数	订阅目标的使用者数量。
EnqueueCount	计数	每分钟发送到目标的消息数量。
EnqueueTime	时间 (毫秒)	从消息到达代理到传递给使用者的端到端延迟。 **Note** EnqueueTime 不会衡量从生产者发送消息到消息到达代理之间的端到端延迟，也不会衡量从代理收到消息到代理确认消息之间的延迟。相反，EnqueueTime 是从代理收到消息到成功传递给使用者的毫秒数。
ExpiredCount	计数	每分钟因过期而无法提供的消息数量。
DispatchCount	计数	每分钟发送到使用者的消息数量。
DequeueCount	计数	每分钟使用者确认的消息数量。
InFlightCount	计数	发送给使用者但尚未确认的消息数量。
ReceiveCount	计数	已从双工网络连接器的远程代理接收的消息数。

指标	Unit	描述
MemoryUsage	百分比	目标位置当前使用的内存限制的百分比。
ProducerCount	计数	目标位置的创建者数量。
QueueSize	计数	队列中的消息数量。 **Important** 此指标仅适用于队列。
TotalEnqueueCount	计数	已发送到代理的消息总数。
TotalDequeueCount	计数	客户端已使用的消息总数。

Note

TotalEnqueueCount 和 TotalDequeueCount 指标包括有关公告主题的消息。有关咨询主题消息的更多信息，请参阅 ActiveMQ 文档。

ActiveMQ 目标（队列和主题）指标的维度

维度	描述
Broker	代理的名称。 **Note** 单实例代理具有后缀 -1。高可用性的主动/备用代理具有后缀 -1 和 -2 可用于其冗余对。
Topic 或者 Queue	主题或队列的名称。

维度	描述
NetworkConnector	网络连接器的名称。

记录和监控 Amazon MQ for RabbitMQ 代理

RabbitMQ 代理指标

指标	Unit	描述
ExchangeCount	计数	在代理上配置的交换器总数。
QueueCount	计数	在代理上配置的队列总数。
ConnectionCount	计数	在代理上建立的连接总数。
ChannelCount	计数	在代理上建立的通道总数。
ConsumerCount	计数	连接到代理的使用者总数。
MessageCount	计数	队列中的消息总数。 **Note** 生成的数字是代理上已就绪和未确认的消息总和。
MessageReadyCount	计数	队列中已就绪的消息总数。
MessageUnacknowledgedCount	计数	队列中未确认的消息总数。
PublishRate	计数	向代理发布消息的速率。 生成的数字表示采样时每秒采集的消息数。

指标	Unit	描述
ConfirmRate	计数	RabbitMQ 服务器确认已发布消息的速率。您可以将此指标与 PublishRate 进行比较，以更好地了解您的代理的表现。 生成的数字表示采样时每秒采集的消息数。
AckRate	计数	使用者确认消息的速率。 生成的数字表示采样时每秒采集的消息数。
SystemCpuUtilization	百分比	代理当前正在使用的已分配 Amazon EC2 计算单位的百分比。对于集群部署，此值表示所有三个 RabbitMQ 节点的相应指标值的总和。
RabbitMQMemLimit	字节	RabbitMQ 代理的 RAM 限制。对于集群部署，此值表示所有三个 RabbitMQ 节点的相应指标值的总和。
RabbitMQMemUsed	字节	RabbitMQ 代理使用的 RAM 容量。对于集群部署，此值表示所有三个 RabbitMQ 节点的相应指标值的总和。

指标	Unit	描述
RabbitMQDiskFreeLimit	字节	RabbitMQ 代理的磁盘限制。对于集群部署，此值表示所有三个 RabbitMQ 节点的相应指标值的总和。该指标因实例大小而异。有关 Amazon MQ 实例类型的更多信息，请参阅 the section called “Amazon MQ for RabbitMQ 实例类型”。
RabbitMQDiskFree	字节	RabbitMQ 代理中可用的免费磁盘空间总量。当磁盘使用量超过其限制时，集群将阻止所有生产者连接。对于集群部署，此值表示所有三个 RabbitMQ 节点的相应指标值的总和。
RabbitMQFdUsed	计数	使用的文件描述符数。对于集群部署，此值表示所有三个 RabbitMQ 节点的相应指标值的总和。
RabbitMQIOReadAverageTime	计数	RabbitMQ 执行一次读取操作的平均时间（以毫秒为单位）。该值与消息大小成正比。
RabbitMQIOWriteAverageTime	计数	RabbitMQ 执行一次写入操作的平均时间（以毫秒为单位）。该值与消息大小成正比。

RabbitMQ 代理指标的维度

维度	描述
Broker	代理的名称。

RabbitMQ 节点指标

指标	Unit	描述
SystemCpuUtilization	百分比	代理当前正在使用的已分配 Amazon EC2 计算单位的百分比。
RabbitMQMemLimit	字节	RabbitMQ 节点的 RAM 限制。
RabbitMQMemUsed	字节	RabbitMQ 节点使用的 RAM 容量。当内存使用量超过限制时，集群将阻止所有生产者连接。
RabbitMQDiskFreeLimit	字节	RabbitMQ 节点的磁盘限制。该指标因实例大小而异。有关 Amazon MQ 实例类型的更多信息，请参阅 the section called “Amazon MQ for RabbitMQ 实例类型”。
RabbitMQDiskFree	字节	RabbitMQ 节点中可用的免费磁盘空间总量。当磁盘使用量超过其限制时，集群将阻止所有生产者连接。
RabbitMQFdUsed	计数	使用的文件描述符数。

RabbitMQ 节点指标的维度

维度	描述
Node	节点的名称。 **Note** 节点名称由两部分组成：前缀（通常为 rabbit）和一个主机名。例如，节点名称 rabbit@ip-10-0-0-230.us-west-2.compute.internal 的前缀为 rabbit，主机名为 ip-10-0-0-230.us-west-2.compute.internal 。
Broker	代理的名称。

RabbitMQ 队列指标

指标	Unit	描述
ConsumerCount	计数	订阅队列的使用者数量。
MessageReadyCount	计数	当前可以传送的消息数量。
MessageUnacknowledgedCount	计数	服务器正在等待确认的消息数量。
MessageCount	计数	MessageReadyCount 和 MessageUnacknowledgedCount 的总数（也称为队列深度）。

RabbitMQ 队列指标的维度

> **Note**
>
> Amazon MQ for RabbitMQ 不会为名称包含空格、制表符或其他非 ASCII 字符的虚拟主机和队列发布指标。
> 有关维度名称的更多信息，请参阅《Amazon CloudWatch API 参考》中的维度。

维度	描述
`Queue`	队列的名称。
`VirtualHost`	虚拟主机的名称。
`Broker`	代理的名称。

使用 AWS CloudTrail 记录 Amazon MQ API 调用

Amazon MQ 与 AWS CloudTrail 集成，该服务提供用户、角色或AWS服务调用 Amazon MQ 的记录。CloudTrail 将与 Amazon MQ 代理和配置相关的 API 调用作为事件捕获，包括来自 Amazon MQ 控制台的调用和来自 Amazon MQ API 的代码调用。有关 CloudTrail 的更多信息，请参阅《AWS CloudTrail 用户指南》。

> **Note**
>
> CloudTrail 不会记录与 ActiveMQ 操作（例如，发送和接收消息）或 ActiveMQ Web 控制台相关的 API 调用。要记录与 ActiveMQ 操作相关的信息，您可以配置 Amazon MQ 以将常规日志和审计日志发布到 Amazon CloudWatch Logs。

使用 CloudTrail 所收集的信息，可以确定向 Amazon MQ API 发出的特定请求、请求者的 IP 地址、请求者的身份、请求的日期和时间等。如果您配置了跟踪，则可以使 CloudTrail 事件持续传送到 Amazon S3 存储桶。如果没有配置跟踪，则可以在 CloudTrail 控制台中的事件历史记录中查看最新事件。有关更多信息，请参阅《AWS CloudTrail 用户指南》https://docs.aws.amazon.com/awscloudtrail/latest/userguide/中的创建跟踪概述。

CloudTrail 中的 Amazon MQ 信息

创建AWS账户时，会启用 CloudTrail。当发生受支持的 Amazon MQ 事件时，该事件会记录在 CloudTrail 事件中，并与其他AWS服务事件一同保存在事件历史记录中。您可以查看、搜索和下载 AWS 账户的最新事件。有关更多信息，请参阅《AWS CloudTrail 用户指南》中的使用 CloudTrail 事件历史记录查看事件。

通过跟踪，CloudTrail 可将日志文件传送至 Amazon S3 存储桶。您可以创建跟踪以持续记录 AWS 账户中的事件。默认情况下，在使用 AWS Management Console创建跟踪时，此跟踪将应用于所有 AWS 区域。此跟踪记录所有AWS区域中的事件，并将日志文件传送至指定的 Amazon S3 存储桶。您还可以配置其他AWS服务，进一步分析在 CloudTrail 日志中收集的事件数据并采取措施。有关更多信息，请参阅 AWS CloudTrail 用户指南 中的以下主题：

- CloudTrail 支持的服务和集成
- 为 CloudTrail 配置 Amazon SNS 通知
- 从多个区域接收 CloudTrail 日志文件
- 从多个账户接收 CloudTrail 日志文件

Amazon MQ 支持将以下 API 的请求参数和响应作为事件记录在 CloudTrail 日志文件中：

- `CreateConfiguration`
- `DeleteBroker`
- `DeleteUser`
- `RebootBroker`
- `UpdateBroker`

> **Note**
>
> 当您重新启动代理时，会记录 RebootBroker 日志文件。在维护窗口期间，服务会自动重新启动，并且不会记录 rebootBroker 日志文件。

> **Important**
>
> 对于以下 API 的 GET 方法，将会记录请求参数，但会掩蔽响应：

- DescribeBroker
- DescribeConfiguration
- DescribeConfigurationRevision
- DescribeUser
- ListBrokers
- ListConfigurationRevisions
- ListConfigurations
- ListUsers

对于以下 API，通过星号 (data) 隐藏了 password 和 *** 请求参数：

- CreateBroker (POST)
- CreateUser (POST)
- UpdateConfiguration (PUT)
- UpdateUser (PUT)

每个事件或日志条目都包含有关请求者的信息。此信息可帮助您确定以下内容：

- 请求是使用根用户凭证还是 用户凭证发出的？
- 请求是使用角色还是联合身份用户的临时安全凭证发出的？
- 请求是否由其他 AWS 服务发出？

有关更多信息，请参阅《AWS CloudTrail 用户指南》中的 CloudTrail userIdentity 元素。

示例 Amazon MQ 日志文件条目

跟踪是一种配置，可用于将事件作为日志文件传送到指定的 Amazon S3 存储桶。CloudTrail 日志文件包含一个或多个日志条目。

事件表示来自任何源的单个请求，其中包括有关对 Amazon MQ API 的请求、请求者的 IP 地址、请求者的身份、请求的日期和时间等的信息。

以下示例说明 CreateBroker API 调用的 CloudTrail 日志条目。

> **Note**
>
> 因为 CloudTrail 日志文件不是公用 API 的有序堆栈跟踪，所以它们不会按任何特定顺序列出信息。


```
{
    "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"
}
```


配置 Amazon MQ 以将日志发布到 Amazon CloudWatch Logs

Amazon MQ 与 Amazon CloudWatch Logs 集成，后者是一项监控、存储和访问来自各种源的日志文件的服务。例如，您可以配置 CloudWatch 警报来接收代理重启的通知，或者解决 ActiveMQ 代理配置错误。有关 CloudWatch Logs 的更多信息，请参阅 Amazon CloudWatch Logs 用户指南。

主题

- 配置 Amazon MQ for ActiveMQ 日志
- 配置 Amazon MQ for RabbitMQ 日志

配置 Amazon MQ for ActiveMQ 日志

要允许 Amazon MQ 将日志发布到 CloudWatch Logs，您必须为 Amazon MQ 用户添加权限以及为 Amazon MQ 配置基于资源的策略，然后再创建或重启代理。

下面介绍了为您的 ActiveMQ 代理配置 CloudWatch Logs 的步骤。

主题

了解 CloudWatch Logs 中的日志记录的结构

在创建代理或编辑代理时，您可以在配置高级代理设置时启用常规和审核日志记录。

常规日志记录启用默认的 `INFO` 日志记录级别（不支持 `DEBUG` 日志记录）并将 `activemq.log` 发布到 CloudWatch 账户中的日志组。日志组的格式如下所示：

```
/aws/amazonmq/broker/b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9/general
```

审核日志记录用于记录使用 JMX 或使用 ActiveMQ Web 控制台执行的管理操作的日志并将 `audit.log` 发布到 CloudWatch 账户中的日志组。日志组的格式如下所示：

```
/aws/amazonmq/broker/b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9/audit
```

根据您是具有单实例代理还是主动/备用代理，Amazon MQ 将在每个日志组中创建一个或两个日志流。日志流的格式如下所示。

```
activemq-b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.log
activemq-b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-2.log
```

`-1` 和 `-2` 后缀表示单个代理实例。有关更多信息，请参阅 Amazon CloudWatch Logs 用户指南 中的使用日志组和日志流。

向 Amazon MQ 用户添加 **CreateLogGroup** 权限

要允许 Amazon MQ 创建 CloudWatch Logs 日志组，您必须确保创建或重启代理程序的用户具有 `logs:CreateLogGroup` 权限。

> **Important**
>
> 如果您未在 Amazon MQ 用户创建或重启代理之前将 `CreateLogGroup` 权限添加给 Amazon MQ 用户，则 Amazon MQ 不会创建日志组。

以下示例介绍基于 IAM 的策略为附加此策略的用户授予 `logs:CreateLogGroup` 权限。

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": "logs:CreateLogGroup",
         "Resource": "arn:aws:logs:*:*:log-group:/aws/amazonmq/*"
      }
   ]
}
```

> **Note**
>
> 此处，术语“用户”是指用户而不是 Amazon MQ 用户，后者是在配置新的代理程序时创建的。有关设置用户和配置 IAM policy 的更多信息，请参阅《IAM 用户指南》中的身份管理概述部分。

有关更多信息，请参阅《Amazon CloudWatch Logs API 参考》中的 `CreateLogGroup`。

为 Amazon MQ 配置基于资源的策略。

> **Important**
>
> 如果您没有为 Amazon MQ 配置基于资源的策略，则代理无法将日志发布到 CloudWatch Logs。

要允许 Amazon MQ 将日志发布到 CloudWatch Logs 日志组，请配置一个基于资源的策略来将 Amazon MQ 访问权限授予以下 CloudWatch Logs API 操作：

- CreateLogStream – 为指定的日志组创建 CloudWatch Logs 日志流。
- PutLogEvents – 将事件传送到指定的 CloudWatch Logs 日志流。

以下基于资源的策略将 `logs:CreateLogStream` 和 `logs:PutLogEvents` 的权限授予 AWS。

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

必须使用 AWS CLI 配置此基于资源的策略，如以下命令所示。在示例中，将 *us-east-1* 替换为您自己的信息。

```
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**
>
> 由于此示例使用 `/aws/amazonmq/` 前缀，您只需为每个区域的每个 AWS 账户配置一次基于资源的策略。

跨服务混淆代理问题防范

混淆代理问题是一个安全性问题，即不具有操作执行权限的实体可能会迫使具有更高权限的实体执行该操作。在 AWS 中，跨服务模拟可能会导致混淆代理问题。一个服务（呼叫服务）调用另一项服务（所

谓的服务)时，可能会发生跨服务模拟。可以操纵调用服务，使用其权限以在其他情况下该服务不应有访问权限的方式对另一个客户的资源进行操作。为了防止这种情况，AWS 提供可帮助您保护所有服务的服务委托人数据的工具，这些服务委托人有权限访问账户中的资源。

我们建议在基于 Amazon MQ 资源的策略中使用 `aws:SourceArn` 和 `aws:SourceAccount` 全局上下文键限制至一个或多个指定代理的 CloudWatch Logs 访问。

> **Note**
>
> 如果使用两个全局条件上下文键，在同一策略语句中使用时，`aws:SourceAccount` 值和 `aws:SourceArn` 值中的账户必须使用相同的账户 ID。

以下示例演示了基于资源的策略，该策略可限制至单个 Amazon MQ 代理的 CloudWatch Logs 访问。

```
{
  "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"
        }
      }
    }
  ]
}
```

此外，您还可以配置基于资源的策略，以限制至账户中的所有代理的 CloudWatch Logs 访问，如以下所示。

```
{
    "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"
          }
        }
      }
    ]
  }
```

有关混淆代理人安全问题的更多信息，请参阅《用户指南》中的混淆代理人问题。

CloudWatch Logs 配置问题排查

在某些情况下，CloudWatch Logs 的行为可能并不总是符合预期。此部分概述了常见问题并说明如何解决这些问题。

日志组没有显示在 CloudWatch 中

将 CreateLogGroup 权限添加给 Amazon MQ 用户并重启代理。这样 Amazon MQ 就可以创建日志组了。

日志流没有显示在 CloudWatch 日志组中

为 Amazon MQ 配置基于资源的策略。这样代理就可以发布其日志了。

配置 Amazon MQ for RabbitMQ 日志

当您为 RabbitMQ 代理启用 CloudWatch 日志记录时，Amazon MQ 会使用服务相关角色将常规日志发布到 CloudWatch。如果您首次创建代理时不存在与 Amazon MQ 服务相关的角色，Amazon MQ 将自动创建一个角色。所有后续 RabbitMQ 代理将使用相同的服务相关角色将日志发布到 CloudWatch。

有关服务相关角色的更多信息，请参阅《AWS Identity and Access Management 用户指南》中的使用服务相关角色。有关 Amazon MQ 如何使用服务相关角色的更多信息，请参阅the section called “使用服务相关角色”。

Amazon MQ 中的配额

本主题列出了 Amazon MQ 中的配额。对于特定 AWS 账户，下面的许多配额可能会发生更改。要请求提高限制，请参阅《Amazon Web Services 一般参考》中的 AWS 服务限额。即使提高限制后，更新后的限制也将不可见。有关在 Amazon CloudWatch 中查看当前连接限制的更多信息，请参阅使用 Amazon CloudWatch 监控 Amazon MQ 代理。

主题

代理

下表列出了与 Amazon MQ 代理相关的配额。

限制	描述
代理名称	• 在您的 AWS 账户中必须是唯一的。 • 长度必须介于 1 到 50 个字符之间。 • 只能包含 ASCII 可打印字符集中指定的字符。 • 只能包含字母数字字符、短划线、句点、下划线和波浪线 (- . _ ~)。
每个区域的代理数	50
小型代理每个协议的线级连接数	**Important** 不适用于 RabbitMQ 代理。

限制	描述
	`mq.*.micro` 实例类型代理为 300。
大型代理每个协议的线级连接数	**Important** 不适用于 RabbitMQ 代理。 `mq.*.*large` 实例类型代理为 2000。
每个代理的安全组	5
在 CloudWatch 中监控的 ActiveMQ 目标（队列和主题）	CloudWatch 只监控前 1000 个目标。
CloudWatch 中监控的 RabbitMQ 目标（队列）	CloudWatch 只监控前 500 个目标，按使用者数量排序。
每个代理的标签数	50

配置

下表列出了与 Amazon MQ 配置相关的配额。

Important
不适用于 RabbitMQ 代理。

限制	描述
配置名称	• 长度必须介于 1 到 150 个字符之间。 • 只能包含 ASCII 可打印字符集中指定的字符。 •

限制	描述
	只能包含字母数字字符、短划线、句点、下划线和波浪线 (- . _ ~)。
每个配置的修订	300

用户

下表列出了与 Amazon MQ ActiveMQ 代理用户相关的配额。

> **Important**
>
> 不适用于 RabbitMQ 代理。

限制	描述
Username	• 长度必须介于 1 到 100 个字符之间。 • 只能包含 ASCII 可打印字符集中指定的字符。 • 只能包含字母数字字符、短划线、句点、下划线和波浪线 (- . _ ~)。 • 不能包含逗号 (,)。
Password	• 长度必须介于 12 到 250 个字符之间。 • 只能包含 ASCII 可打印字符集中指定的字符。 • 必须至少包含 4 个唯一字符。 • 不能包含逗号 (,)。
每个代理的用户（简单身份验证）	250

限制	描述
每个用户的组数（简单身份验证）	20

数据存储

下表列出了与 Amazon MQ 数据存储相关的配额。

限制	描述
每个小型代理的存储容量	`mq.*.micro` 实例类型代理为 20GB。有关 Amazon MQ 实例类型的更多信息，请参阅Broker instance types。
每个大型代理的存储容量	`mq.*.*large` 实例类型代理为 200GB。有关 Amazon MQ 实例类型的更多信息，请参阅Broker instance types。
Amazon EBS 支持的每个代理的任务计划程序使用限制	**Important** 不适用于 RabbitMQ 代理。 50GB。有关任务计划程序使用的更多信息，请参阅《Apache ActiveMQ API 文档》中的`JobSchedulerUsage` 。
每个小型代理的临时存储容量	**Important** 不适用于 RabbitMQ 代理。 `mq.*.micro` 实例类型代理为 5GB。
每个大型代理的临时存储容量	

限制	描述
	Important 不适用于 RabbitMQ 代理。 `mq.*.*large` 实例类型代理为 50GB。

API 限制

以下节流配额按AWS账户跨所有 Amazon MQ API 聚合，以维护服务带宽。要了解有关 Amazon MQ API 的更多信息，请参阅 Amazon MQ REST API 参考。

Important
这些配额不适用于 Amazon MQ for ActiveMQ 或 Amazon MQ for RabbitMQ 代理消息 API。例如，Amazon MQ 不会限制消息的发送或接收。

API 突增限制	API 速率限制
100	15

Amazon MQ 问题排查

本节将介绍使用 Amazon MQ 代理时可能遇到的常见问题，以及解决问题的步骤。

目录

- 适用于 RabbitMQ 的亚马逊 MQ：密钥无效 AWS Key Management Service
  - 诊断和解决 INVALID_KMS_KEY
- Amazon MQ for ActiveMQ：已删除弹性网络接口警报
- Amazon MQ for ActiveMQ：代理内存不足警报
- Amazon MQ for RabbitMQ：磁盘限制警报
  - 诊断和解决磁盘限制警报

问题排查：一般问题

使用本节中的信息帮助您诊断在使用 Amazon MQ 代理时可能遇到的常见问题，例如连接到代理的问题和代理重启。

目录

我无法连接到代理 Web 控制台或终端节点。

如果您在使用 Web 控制台或线级终端节点连接到代理时遇到问题，我们建议按以下步骤操作。

1. 检查您是否尝试从防火墙后面连接到您的代理。您可能需要配置防火墙以允许访问您的代理。
2. 检查您是否正在尝试使用 FIPS 端点连接代理。Amazon MQ 仅在使用 API 操作时支持 FIPS 端点，而不支持与代理实例本身的线级连接。
3. 检查代理的 Public Accessibility (公开可访问性) 选项是否设置为 Yes (是)。如果设置为 No (否)，请检查您的子网网络访问控制列表（ACL）规则。如果您已创建自定义网络 ACL，则可能需要更改网络 ACL 规则以提供对代理的访问权限。有关 Amazon VPC 联网的更多信息，请参阅《Amazon VPC 用户指南》中的启用互联网访问
4. 检查您的代理的安全组规则。确保您允许连接到以下端口：

> **Note**
>
> 以下端口根据引擎类型进行分组，因为 Amazon MQ for ActiveMQ 和 Amazon MQ for RabbitMQ 使用不同的端口进行连接。

Amazon MQ for ActiveMQ

- Web 控制台 – 端口 8162
- OpenWire — 端口 61617
- AMQP – 端口 5671
- STOMP — 端口 61614
- MQTT – 端口 8883
- WSS – 端口 61619

Amazon MQ for RabbitMQ

- Web 控制台和管理 API – 端口 443 和 15671
- AMQP – 端口 5671

5. 为您的代理引擎类型运行以下网络连接测试。

> **Note**
>
> 对于不可公开访问的代理，请在与 Amazon MQ 代理相同的 Amazon VPC 中从 Amazon EC2 实例运行测试并评估响应。

Amazon MQ for ActiveMQ

测试您的 Amazon MQ for ActiveMQ 代理的网络连接

1. 打开新终端或命令行窗口。
2. 运行以下 `nslookup` 命令，查询您的代理 DNS 记录。对于主/备用部署，请同时测试主备用终端节点。主/备用终端节点以添加到唯一代理 ID 的后缀、`-1` 或 `-2` 标识。将终端节点替换为您的信息。

```
$ nslookup b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-west-2.amazonaws.com
```

如果查询成功，则将显示类似于以下内容的输出。

```
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
```

已解析的 IP 地址应与 Amazon MQ 控制台中提供的 IP 地址相匹配。这表示域名在 DNS 服务器上正确解析，您可以继续执行下一步。

3. 运行以下 `telnet` 命令，测试您的代理的网络路径。将终端节点替换为您的信息。将`##`替换为 Web 控制台的端口号 8162，或根据需要替换为其他线级端口以测试更多协议。

> **Note**
>
> 对于主/备用部署，如果您通过备用终端节点运行 `telnet`，则将收到 `Connect failed` 错误消息。这在预期之内，因为备用实例本身正在运行，但 ActiveMQ 进程没有运行，并且未拥有访问代理的 Amazon EFS 存储卷的权限。对 `-1` 和 `-2` 终端节点运行该命令，以确保同时测试主备用实例。

```
$ telnet b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-
west-2.amazonaws.com port
```

对于主动实例，将显示类似于以下内容的输出。

```
Connected to b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-
west-2.amazonaws.com.
Escape character is '^]'.
```

4. 请执行以下操作之一。

   - 如果 `telnet` 命令成功，请检查 `EstablishedConnectionsCount` 指标，并确认代理未达到线级连接上限。您还可以通过查看代理 `General` 日志来确认是否达到上限。如果

此指标大于零，则目前至少有一个客户端连接到该代理。如果指标显示零连接，则再次进行 telnet 路径测试，并等待至少一分钟才能断开连接，因为代理指标每分钟发布一次。

- 如果 telnet 命令失败，请检查代理的弹性网络接口，并确认状态为 in-use。为每个实例的网络接口创建 Amazon VPC 流日志，然后查看生成的流日志。查找运行 telnet 命令时的代理 IP 地址，并确认连接数据包是 ACCEPTED，包括返回数据包。要了解更多信息和查看流日志示例，请参阅《Amazon VPC 开发人员指南》中的流日志目录示例。

5. 运行以下 curl 命令，检查 ActiveMQ 管理 Web 控制台的连接。

```
$ curl https://b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-
west-2.amazonaws.com:8162/index.html
```

如果命令成功，则输出应是类似于以下内容的 HTML 文档。

```
<!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

测试您的 Amazon MQ for RabbitMQ 代理的网络连接

1. 打开新终端或命令行窗口。
2. 运行以下 nslookup 命令，查询您的代理 DNS 记录。将终端节点替换为您的信息。

```
$ nslookup b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-west-2.amazonaws.com
```

如果查询成功，则将显示类似于以下内容的输出。

```
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. 运行以下 telnet 命令，测试您的代理的网络路径。将终端节点替换为您的信息。您可以将 *port* 替换为 Web 控制台的端口 443，并替换为 5671 以测试线级 AMQP 连接。

```
$ telnet b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-
west-2.amazonaws.com port
```

如果命令成功，则将显示类似于以下内容的输出。

```
Connected to b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-
west-2.amazonaws.com.
Escape character is '^]'.
```

> **Note**
>
> Telnet 连接将在几秒钟后自动关闭。

4. 请执行以下操作之一。

- 如果 telnet 命令成功，请检查 ConnectionCount 指标，并确认代理未达到 max-connections 默认策略中设置的值。您还可以通过查看代理 Connection.log 日志组来确认是否达到上限。如果此指标大于零，则目前至少有一个客户端连接到该代理。如果指标显示零连接，则再次进行 telnet 路径测试。如果在您的代理向其发布新的连接指标之前连接关闭，则可能需要重复此过程 CloudWatch。每分钟发布一次指标。
- 对于不可公开访问的代理，如果 telnet 命令失败，请检查代理的弹性网络接口，并确认状态为 in-use。为每个网络接口创建 Amazon VPC 流日志，然后查看生成的流日志。查找调用 telnet 命令时的代理 IP 地址，并确认连接数据包是 ACCEPTED，包括返回数据包。要了解更多信息和查看流日志示例，请参阅《Amazon VPC 开发人员指南》中的流日志目录示例。

> **Note**
>
> 此步骤不适用于可公开访问的 Amazon MQ for RabbitMQ 代理。

5. 运行以下 `curl` 命令，检查 RabbitMQ 管理 Web 控制台的连接。

```
$ curl https://b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-
west-2.amazonaws.com:443/index.html
```

如果命令成功，则输出应是类似于以下内容的 HTML 文档。

```
<!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>
        ...
```

我的代理正在运行，我可以使用 **telnet** 验证连接情况，但是我的客户端无法连接并且正在返回 SSL 异常情况。

您的代理端点证书可能已在代理维护时段期间更新。Amazon MQ 代理证书会定期轮换，以确保代理的持续可用性和安全性。

我们建议在 Amazon Trust Services 中使用 Amazon 根证书颁发机构 (CA)，在您客户端的信任存储中进行身份验证。所有 Amazon MQ 代理证书均使用此根 CA 签名。通过使用 Amazon 根 CA，您不再需要每次在代理上更新证书时下载新的 Amazon MQ 代理证书。

我创建了一个代理，但代理创建失败。

如果您的代理处于 `CREATION_FAILED` 状态，请执行以下操作。

- 检查您的 IAM 权限。要创建代理，必须使用 AWS 托管 IAM 策略`AmazonMQFullAccess`或在自定义 IAM 策略中拥有正确的 Amazon EC2 权限集。有关您需要的 Amazon EC2 权限的更多信息，请参阅创建 Amazon MQ 代理时所需的 IAM 权限。

- 检查您为代理选择的子网是否位于共享的 Amazon Virtual Private Cloud（VPC）中。要在共享的 Amazon VPC 中创建 Amazon MQ 代理，您必须在拥有 Amazon VPC 的账户中创建它。

我的代理重启，我不知道原因是什么。

如果代理已自动重启，则可能是由以下某个原因所致。

- 您的代理之所以重启，可能是因为到了每周计划维护时段。Amazon MQ 定期对消息代理的硬件、操作系统或引擎软件进行维护。维护的持续时间有所不同，但最多可持续两小时，具体取决于为消息代理安排的操作。代理可能会在两小时维护时段内的任何时间点重启。有关代理维护时段的更多信息，请参阅 the section called “维护代理”。
- 您的代理实例类型可能不适合您的应用程序工作负载。例如，在 `mq.t2.micro` 上运行生产工作负载可能会导致代理耗尽资源。较高的 CPU 利用率或较高的代理内存使用率可能会导致代理意外重启。要查看您的代理使用了多少 CPU 和内存，请根据您的引擎类型使用以下 CloudWatch 指标。
  - Amazon MQ for ActiveMQ – 检查 `CpuUtilization`，了解代理目前使用的已分配 Amazon EC2 计算单位的百分比。检查 `HeapUsage`，了解代理目前使用的 ActiveMQ JVM 内存限制的百分比。
  - Amazon MQ for RabbitMQ – 检查 `SystemCpuUtilization`，了解代理目前使用的已分配 Amazon EC2 计算单位的百分比。检查 `RabbitMQMemUsed`，了解以字节为单位的 RAM 使用量，然后除以 `RabbitMQMemLimit`，得出 RabbitMQ 节点使用的内存百分比。

  有关代理实例类型以及如何为工作负载选择合适的实例类型的更多信息，请参阅 Broker instance types。

问题排查：Amazon MQ for ActiveMQ

使用本节中的信息帮助您诊断和解决在使用 Amazon MQ for ActiveMQ 代理时可能遇到的常见问题。

目录

尽管我已经激活了日志记录，但我也无法在 CloudWatch 日志中看到我的经纪人的常规日志或审计日志。

如果您无法在 Logs 中查看经纪人的 CloudWatch 日志，请执行以下操作。

1. 检查创建或重启代理程序的用户是否具有 `logs:CreateLogGroup` 权限。如果您未在用户创建或重启代理之前将 `CreateLogGroup` 权限添加给该用户，则 Amazon MQ 不会创建日志组。
2. 检查您是否已将基于资源的策略配置为允许 Amazon MQ 将日志发布到日志 CloudWatch 。要允许 Amazon MQ 将日志发布到您的日志 CloudWatch 日志组，请配置基于资源的策略以授予 Amazon MQ 访问以下日志 API 操作的权限： CloudWatch
   - `CreateLogStream`— 为指定的 CloudWatch 日志组创建日志日志流。
   - `PutLogEvents`— 将事件传送到指定的 CloudWatch 日志日志流。

有关配置 Amazon MQ for ActiveMQ 以将日志发布到日志的更多信息，请参阅配置日志。 CloudWatch

代理重启或维护时段后，即使状态为 **RUNNING**，我也无法连接到我的代理。为什么？

您可能会在您启动的代理重启后、计划的维护时段完成后或在故障事件中遇到连接问题，此时备用实例会激活。无论是哪种情况，代理重启后的连接问题很可能是由于在您代理的 Amazon EFS 或 Amazon EBS 存储卷中保留了异常大量的消息引起的。在重启期间，Amazon MQ 会将储存的消息从存储移动到代理内存。要确认此诊断，您可以监控 CloudWatch 适用于 ActiveMQ 的 Amazon MQ 代理商的以下指标：

- **`StoragePercentUsage`** - 达到或接近 100% 的大百分比会导致代理拒绝连接。
- **`JournalFilesForFullRecovery`** - 表示在不正常关闭和重启后将重播的日志文件数。值增加或持续高于 1 表示存在未解决的事务，可能会在重启后导致连接问题。
- **`OpenTransactionCount`** - 重启后的数字大于零表示代理将尝试存储以前使用的消息，从而导致连接问题。

要解决此问题，我们建议您使用 `rollback()` 或 `commit()` 来解决 XA 事务。有关更多信息，以及要查看使用 `rollback()` 解决 XA 事务的代码示例，请参阅恢复 XA 事务。

我看到我的一些客户端可以连接到代理，而其他客户端则无法连接。

如果您的代理处于 RUNNING 状态并且一些客户端能够成功连接到代理，而其他客户端无法连接，您可能已经达到了代理的线级连接数限制。要验证您是否已达到线级连接数限制，请执行以下操作：

- 在日志中查看您的 Amazon MQ for ActiveMQ 代理的通用经纪商日志。 CloudWatch 如果已达到限制，您会在代理日志中看到 `Reached Maximum Connections`。有关适用于 ActiveMQ 代理的 Amazon MQ CloudWatch 日志的更多信息，请参阅。the section called “了解 CloudWatch Logs 中的日志记录的结构”

一旦达到线级连接数限制，代理将主动拒绝额外的传入连接。要解决此问题，我们建议升级您的代理实例类型。有关为工作负载选择最佳实例类型的更多信息，请参阅Broker instance types。

如果您确认线级连接数低于代理的连接数限制，则问题可能与重新启动客户端有关。检查您的代理日志，查找大量且重复出现的 `... Inactive for longer than 600000 ms - removing ...` 条目。日志条目能够表明客户端重启或连接问题。当客户端通过 Network Load Balancer (NLB) 连接到代理，并经常断开连接而重新连接到代理时，这种迹象就会更加明显。这在基于容器的客户端中更为常见。

请检查客户端日志，了解更多详细信息。代理将在 600000 毫秒后清理不活动的 TCP 连接，并释放连接套接字。

我在执行操作时，在 ActiveMQ 控制台上看到 **`org.apache.jasper.JasperException: An exception occurred processing JSP page`** 异常。

如果您在授权队列和主题时使用简单的身份验证和配置 `AuthorizationPlugin`，请确保使用 XML 配置文件中的 `AuthorizationEntries` 元素，并允许 `activemq-webconsole` 群组拥有所有队列和主题的权限。这可以确保 ActiveMQ Web 控制台能够与 ActiveMQ 代理进行通信。

以下示例 `AuthorizationEntry` 将所有队列和主题的读取和写入权限授予 `activemq-webconsole` 群组。

```
<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>
```

同样，当您的代理与 LDAP 集成时，请确保将权限授予 amazonmq-console-admins 群组。有关 LDAP 集成的更多信息，请参阅 the section called “LDAP 集成的工作方式”。

问题排查：Amazon MQ for RabbitMQ

使用本节中的信息帮助您诊断和解决在使用 Amazon MQ for RabbitMQ 代理时可能遇到的常见问题。

目录

我在中看不到我的队列或虚拟主机的指标 CloudWatch。

如果您无法查看队列或虚拟主机的指标 CloudWatch，请检查您的队列或虚拟主机名是否包含任何空格、制表符或其他非 ASCII 字符。

Amazon MQ 无法为名称包含空格、制表符或其他非 ASCII 字符的虚拟主机和队列发布指标。

有关维度名称的更多信息，请参阅 Amazon CloudWatch API 参考中的维度。

在 Amazon MQ for RabbitMQ 中如何启用插件？

Amazon MQ for RabbitMQ 目前仅支持 RabbitMQ 管理、shovel、联合身份验证、一致性哈希交换插件，这些插件在预设情况下都是启用的。有关使用受支持插件的更多信息，请参阅 the section called “插件”。

我无法更改代理的 Amazon VPC 配置。

Amazon MQ 不支持在创建代理后更改 Amazon VPC 配置。请注意，您需要使用新的 Amazon VPC 配置创建新的代理，然后使用新的代理连接 URL 更新客户端连接 URL。

故障排除：Amazon MQ 操作所需代码

如果代理处于运行不良状态，并且需要一组操作才能恢复到正常运行状态，则 Amazon MQ 对于某些 API 操作返回异常，例如 `RebootBroker`。异常包括特定操作所需代码，这些代码可帮助您确定根源、解决问题和让代理恢复正常。

使用以下主题列表标识您收到的操作所需代码，并详细了解我们为解决您的问题而推荐的步骤。

操作所需代码

- Amazon MQ for RabbitMQ：高内存警报
- 适用于 RabbitMQ 的亚马逊 MQ：密钥无效 AWS Key Management Service
- Amazon MQ for ActiveMQ：已删除弹性网络接口警报
- Amazon MQ for ActiveMQ：代理内存不足警报
- Amazon MQ for RabbitMQ：磁盘限制警报

Amazon MQ for RabbitMQ：高内存警报

当代理的内存使用量（由 CloudWatch 指标`RabbitMQMemUsed`标识）超过由标识的内存限制时，RabbitMQ 将发出高内存警报。`RabbitMQMemLimit` `RabbitMQMemLimit`由 Amazon MQ 设置，并已根据每种主机实例类型的可用内存进行了专门调整。

发出高内存警报的 Amazon MQ for RabbitMQ 代理将会阻止所有客户端发布消息。由于内存使用率高，您的代理可能还会遇到其他使警报诊断和解决复杂化的问题。

因内存使用率高而无法完成启动的单实例代理可能会遇到重新启动循环，在此过程中，与代理的交互将会受到限制。在集群部署中，队列可能会遇到不同节点上的副本之间的消息同步暂停问题。队列同步暂停将会阻止消息消耗队列，并且必须在解决内存警报问题时单独解决此问题。

Amazon MQ 在遇到高内存警报时不会重新启动代理，并且将会在该代理继续发出警报时返回 `RebootBroker` API 操作异常。

使用本部分中的信息帮助您诊断和解决代理发出的 RabbitMQ 高内存警报。

> **Note**
>
> 在您采取必要的操作后，RABBITMQ_MEMORY_ALARM 状态可能需要几个小时才能清除。

> **Note**
>
> 您不能将代理程序从 `mq.m5.` 实例类型降级为 `mq.t3.micro` 实例类型。如果您想降级，则必须删除代理程序并创建新的代理程序。

主题

使用 RabbitMQ Web 控制台诊断高内存警报

RabbitMQ Web 控制台可以生成和显示每个节点的详细内存使用情况信息。您可通过以下操作查找此信息：

1. 登录 AWS Management Console 并打开您的经纪商的 RabbitMQ 网页控制台。
2. 在 RabbitMQ 控制台上的 Overview（概览）页面，从 Nodes（节点）列表中选择一个节点名称。
3. 在节点详细信息页面，选择 Memory details（内存详细信息）展开此部分以查看节点的内存使用情况信息。

RabbitMQ 在 Web 控制台中提供的内存使用情况信息可以帮助您确定哪些资源可能消耗过多资源并造成高内存警报。有关 RabbitMQ Web 控制台中可用的内存使用情况详细信息的更多信息，请参阅 RabbitMQ 服务器文档网站上的关于内存使用的推理。

使用 Amazon MQ 指标诊断高内存警报

默认情况下，Amazon MQ 将会为您的代理启用指标。您可以通过访问 CloudWatch 控制台或使用 CloudWatch API 来查看您的经纪商指标。以下指标在诊断 RabbitMQ 高内存警报时非常有用。

亚马逊 MQ 指标 CloudWatch	内存使用过高的原因	
`MessageCount`	消息在被使用或丢弃之前一直存储在内存中。高消息计数可能指示资源过度使用，并且可能导致高内存警报。	

亚马逊 MQ 指标 CloudWatch	内存使用过高的原因	
QueueCount	队列存储在内存中，并且大量队列可能会导致高内存警报。	
ConnectionCount	客户端连接使用内存，并且过多同时连接可能会导致高内存警报。	
ChannelCount	与连接类似，使用每个连接建立的通道也会存储在节点内存中，并且大量通道可能会导致高内存警报。	
ConsumerCount	对于连接至代理的每个使用者，一组消息在传输至使用者之前将从存储加载到内存。大量使用者连接可能造成高内存使用，并导致高内存警报。	
PublishRate	发布消息将会使用代理的内存。如果消息发布到代理的速率过高并且大幅超过代理将消息传输到使用者的速率，则代理可能会遇到高内存警报。	

解决高内存警报

对于您确定的每个因素，我们建议您采取以下操作来缓解和解决代理的高内存警报。

内存使用过高的原因	Amazon MQ 建议	
队列中的消息数过高。	执行以下任一操作： • 使用发布到队列的消息。 • 清除队列中的消息。 • 删除代理中的队列。	

内存使用过高的原因	Amazon MQ 建议
在代理上配置的队列数过高。	减少队列数。
在代理上建立的连接数过高。	减少连接数。有关更多信息，请参阅 the section called “减少连接和通道的数量”。
在代理上建立的通道数过高。	减少通道数。有关更多信息，请参阅 the section called “减少连接和通道的数量”。
连接到代理的使用者数过高。	减少连接到代理的使用者数。
消息发布速率过高。	降低发布商向代理发送消息的速率。
客户端连接尝试率过高。	降低客户端尝试连接到代理以便发布或使用消息或者配置代理的频率。

减少连接和通道的数量

至 Amazon MQ for RabbitMQ 代理的连接可通过客户端应用程序关闭，或者通过使用 RabbitMQ Web 控制台手动将其关闭。要使用 RabbitMQ Web 控制台关闭连接，请执行以下操作。

1. 登录 AWS Management Console 并打开您的经纪商的 RabbitMQ 网页控制台。
2. 在 RabbitMQ 控制台上，选择 Connections（连接）选项卡。
3. 在 Connections（连接）页面的 All connections（所有连接）下，选择您想要从列表中关闭的连接名称。
4. 在连接详细信息页面上，选择 Close this connection（关闭此连接）以展开此部分，然后选择 Force Close（强制关闭）。或者，您可以将 Reason（原因）的默认文本替换为您自己的描述。在您关闭连接时，Amazon MQ for RabbitMQ 将会向客户端返回您指定的原因。
5. 选择对话框上的 OK（确定）以确认并关闭连接。

在您关闭连接时，与关闭的连接关联的任何通道也会关闭。

> **Note**
>
> 您的客户端应用程序可配置为在关闭连接后自动重新建立至代理的连接。在此情况下，从代理 Web 控制台关闭连接可能不足以减少连接或通道计数。

对于没有公共访问的代理，您可以通过拒绝相应消息协议端口（例如，AMQP 连接的端口 5671）上的入站流量来临时阻止连接。创建代理时，您可以阻止您向 Amazon MQ 提供的安全组中的端口。有关修改安全组的更多信息，请参阅 Amazon VPC 用户指南中的向安全组添加规则。

解决集群部署中的队列同步暂停问题

解决 RabbitMQ 的高内存警报时，您可能会发现无法使用一个或多个队列上的消息。这些队列可能正处于节点之间消息同步过程中，在此过程中，相应的队列可能无法用于发布和使用。队列同步可能因高内存警报暂停，甚至可能造成内存警报。

有关停止和重试暂停队列同步的信息，请参阅 the section called “解决暂停队列同步的问题”。

解决单实例代理中的重新启动循环问题

如果发出高内存警报的 Amazon MQ for RabbitMQ 单实例代理重新启动且没有足够的内存启动，则可能会存在不可用风险。这可能会导致 RabbitMQ 进入重新启动玄幻，并且在问题解决之前阻止与代理的任何进一步交互。如果您的代理处在重新启动循环中，则您将无法应用本部分前面描述的 Amazon MQ 建议操作，以解决高内存警报。

要恢复您的代理，我们建议升级到具有更高内存的更大实例类型。与在集群部署中不同，您可以在遇到高内存警报时升级单实例代理，因为在重新启动过程中节点之间没有任何要执行的队列同步。

防止高内存警报

对于您确定的每个起作用的因素，我们建议执行以下操作，以防止和减少 RabbitMQ 高内存警报的发生。

高内存使用率原因	Amazon MQ 建议	
队列中的消息数过高。	执行以下操作： • 启用延迟队列。 • 设置或降低队列深度限制。	

高内存使用率原因	Amazon MQ 建议
在代理上配置的队列数过高。	设置或降低队列计数限制。
在代理上建立的连接数过高。	设置或降低连接计数限制。
在代理上建立的通道数过高。	设置客户端应用程序上的每个连接的最大通道数。
连接到代理的使用者数过高。	设置一个小的使用者预提取限制。
客户端连接尝试率过高。	使用长期连接来降低连接尝试的数量和频率。

代理的内存警报解决之后，您可以将主机实例类型升级为包含其他资源的实例。有关如何更新代理的实例类型的信息，请参阅 Amazon MQ REST API 参考中的 `UpdateBrokerInput`。

有关代理实例类型的完整列表，请参阅 the section called “Amazon MQ for RabbitMQ 实例类型”。

适用于 RabbitMQ 的亚马逊 MQ：密钥无效 AWS Key Management Service

当使用客户托管 AWS KMS key(CMK) 创建的代理检测到 (KMS) 密钥被禁用时，适用于 RabbitMQ 的 Amazon MQ 将引发一个 INVALID_KMS_KEY 关键操作所需的代码。 AWS Key Management Service 拥有 CMK 的 RabbitMQ 代理程序会定期验证 KMS 密钥是否已启用以及代理程序是否具有所有必要的授权。如果 RabbitMQ 无法验证密钥是否已启用，则代理程序将被隔离，RabbitMQ 将返回 INVALID_KMS_KEY。

如果没有有效的 KMS 密钥，代理程序就没有客户托管式 KMS 密钥的基本权限。在您重新启用密钥和代理程序重新启动之前，代理程序无法使用您的密钥执行加密操作。KMS 密钥已禁用的 RabbitMQ 代理程序会被隔离，以防止情况恶化。在 RabbitMQ 确定 KMS 密钥再次处于活动状态后，您的代理程序将从隔离区中移除。Amazon MQ 不会使用禁用的 KMS 密钥重新启动代理程序，并且，只要代理程序继续具有无效的 KMS 密钥，就会为 `RebootBroker` API 操作返回异常。

诊断和解决 INVALID_KMS_KEY

要诊断和解决 INVALID_KMS_KEY 操作所需的代码，必须使用命令 AWS 行界面 (CLI) 和控制台。AWS Key Management Service

重新启用您的 KMS 密钥

1. 调用 `DescribeBroker` 方法以检索 CMK 代理程序的 `kmsKeyId`。
2. 登录 AWS Key Management Service 控制台。
3. 在客户托管式密钥页面上，找到有问题的代理程序的 KMS 密钥 ID，并验证状态为已启用。
4. 如果您的 KMS 密钥已被禁用，请通过选择密钥操作来重新启用密钥，然后选择启用。重新启用密钥后，您必须等待 RabbitMQ 将代理程序从隔离区中删除。

要验证必要的授权是否仍与代理的 KMS 密钥相关联，请调用`ListGrant`ListGrant 方法来验证`mq_rabbit_grant`和`mq_grant`是否存在。如果 KMS 授权或密钥已被删除，则必须删除代理程序，并使用所有必要的授权创建一个新的代理程序。有关删除代理程序的步骤，请参阅删除代理程序。

要防止 INVALID_KMS_KEY 关键操作所需的代码，请勿手动删除或禁用 KMS 密钥或 CMK 授权。如果您想删除密钥，请先删除代理。

Amazon MQ for ActiveMQ：已删除弹性网络接口警报

当您删除代理的弹性网络接口（ENI）时，Amazon MQ for ActiveMQ 将引发 BROKER_ENI_DELETED 警报。在您首次创建 Amazon MQ 代理时，Amazon MQ 会在您账户下的 Virtual Private Cloud（VPC）中预置弹性网络接口，因此需要大量 EC2 权限。

您不得修改或删除此网络接口。修改或删除网络接口可能会导致永久丢失您的 VPC 和代理之间的连接。如果您想删除网络接口，则必须先删除代理。

Amazon MQ for ActiveMQ：代理内存不足警报

当代理由于内存容量不足而陷入重新启动循环时，Amazon MQ for ActiveMQ 将引发 BROKER_OOM 警报。当代理处于重新启动循环（也称为反弹循环）时，代理会在很短的时间窗口内发起反复的恢复尝试。由于内存容量不足而无法完成启动的代理可能会进入重新启动循环，在此过程中，与代理的交互将会受到限制。

默认情况下，Amazon MQ 将会为您的代理启用指标。您可以通过访问 Amazon CloudWatch 控制台或使用 CloudWatch API 来查看您的经纪商指标。诊断 ActiveMQ BROKER_OOM 警报时，以下指标很有用：

亚马逊 MQ 指标 CloudWatch	内存使用过高的原因	
`TotalMessageCount`	消息在被使用或丢弃之前一直存储在内存中。高消息计数可	

亚马逊 MQ 指标 CloudWatch	内存使用过高的原因	
	能指示资源过度使用，并且可能导致高内存警报。	
HeapUsage	代理当前使用的 ActiveMQ JVM 内存限制的百分比。较高的百分比表示代理正在使用大量资源，而可能导致 OOM 警报。	
ConnectionCount	客户端连接使用内存，并且过多同时连接可能会导致高内存警报。	
CpuUtilization	代理当前正在使用的已分配 EC2 计算单位的百分比。	
TotalConsumerCount	对于连接至代理的每个使用者，一组消息在传输至使用者之前将从存储加载到内存。大量使用者连接可能造成高内存使用，并导致高内存警报。	

为防止重新启动循环并避免 BROKER_OOM 警报，请确保快速使用消息。为此，您可以选择最有效的代理实例类型，还可以清除死信队列以丢弃无法传送或过期的消息。您可以在 Amazon MQ for ActiveMQ 最佳实践中详细了解如何确保有效的性能。

Amazon MQ for RabbitMQ：磁盘限制警报

磁盘限制警报表明，由于添加新消息时未消耗大量消息，RabbitMQ 节点使用的磁盘量有所减少。当代理的可用磁盘空间（由 Amazon CloudWatch 指标`RabbitMQDiskFree`标识）达到由标识的磁盘限制时，RabbitMQ 将发出磁盘限制警报。`RabbitMQDiskFreeLimit` `RabbitMQDiskFreeLimit`由 Amazon MQ 设置，定义时考虑了每种代理实例类型的可用磁盘空间。

引发磁盘限制警报的 Amazon MQ for RabbitMQ 代理将变得不可用于所发布的新消息。在集群中运行 RabbitMQ 时，磁盘警报是集群范围的。如果一个节点变得低于限制，则所有其他节点都将阻止传入的消息。由于缺少磁盘空间，代理可能还会遇到其他使警报诊断和解决复杂化的问题。

Amazon MQ 在遇到磁盘警报时不会重新启动代理，并且将会在该代理继续引发警报时返回 `RebootBroker` API 操作异常。

> **Note**
>
> 您不能将代理从 `mq.m5` 实例类型降级为 `mq.t3.micro` 实例类型。如果您想降级，则必须删除代理并创建新的代理。

诊断和解决磁盘限制警报

默认情况下，Amazon MQ 将会为您的代理启用指标。您可以通过访问 Amazon CloudWatch 控制台或使用 CloudWatch API 来查看您的经纪商指标。 `MessageCount`是诊断 RabbitMQ 磁盘限制警报时的有用指标。消息在被使用或丢弃之前一直存储在内存中。高消息计数表示磁盘存储空间过度使用，而可能导致磁盘警报。

要诊断磁盘限制警报，请使用 Amazon MQ 管理控制台执行以下操作：

- 使用发布到队列的消息。
- 清除队列中的消息。
- 删除代理中的队列。

> **Note**
>
> 在您采取必要的操作后，RABBITMQ_DISK_ALARM 状态可能需要几个小时才能清除。

为防止磁盘限制警报再次发生，您可以将主机实例类型升级为具有其他资源的实例。有关如何更新代理的实例类型的信息，请参阅《Amazon MQ REST API 参考》中的 `UpdateBrokerInput`。

相关资源

Amazon MQ 资源

下表列出了关于使用 Amazon MQ 的有用资源。

资源	描述
Amazon MQ REST API 参考	REST 资源、示例请求、HTTP 方法、架构、参数和服务返回的错误的描述。
《AWS CLI 命令参考》中的 Amazon MQ	您用来与消息代理结合使用的 AWS CLI 命令的描述。
《AWS CloudFormation 用户指南》中的 Amazon MQ	通过 `AWS::Amazon MQ::Broker` 资源，您可以创建 Amazon MQ 代理，添加配置更改或修改指定代理的用户，返回有关指定代理的信息以及删除指定代理。 通过 `AWS::Amazon MQ::Configuration` 资源，您可以创建 Amazon MQ 配置，添加配置更改或修改用户，以及返回有关指定配置的信息。
区域和终端节点	有关 Amazon MQ 区域和终端节点的信息
产品页面	提供了 Amazon MQ 相关信息的主要 Web 页面。
开发论坛	一个基于社区的论坛，供开发人员讨论与 Amazon MQ 相关的技术问题。
AWS Premium Support 信息	提供有关 AWS Premium Support 信息的主要 Web 页面，它是一种一对一、快速响应的支持渠道，可帮助您在 AWS 基础设施服务上构建和运行应用程序。

Amazon MQ for ActiveMQ 资源

下表列出了处理 Apache ActiveMQ 的有用资源。

资源	描述
Apache ActiveMQ 入门指南	Apache ActiveMQ 的官方文档。
ActiveMQ 实战	Apache ActiveMQ 的指南，涵盖 JMS 消息、连接器、消息持久性、身份验证和授权的剖析情况。
跨语言客户端	编程语言及对应的 Apache ActiveMQ 库的列表。另请参阅 ActiveMQ 客户端和 QpidJMS 客户端。

Amazon MQ for RabbitMQ 资源

下表列出了关于使用 RabbitMQ 的有用资源。

资源	描述
RabbitMQ 入门指南	RabbitMQ 的官方文档。
RabbitMQ 客户端库和开发工具	官方支持的客户端库和开发工具指南，用于通过各种编程语言和平台使用 RabbitMQ。
RabbitMQ 最佳实践	CloudAMQP 关于使用 RabbitMQ 的最佳实践和建议指南。

Amazon MQ 发布说明

下表列出了 Amazon MQ 特征发布和改进。有关对《Amazon MQ 开发人员指南》的更改，请参阅Amazon MQ 文档历史记录。

Date	文档更新
2024年6月10日	Amazon MQ 现已在加拿大西部（卡尔加里）地区上市。有关可用区域的信息，请参阅《AWS 一般参考》中的 AWS 区域和端点。
2024年5月10日	Amazon MQ 版本支持日历会显示代理引擎版本何时终止支持。当引擎版本终止支持时，Amazon MQ 会自动将该版本上的所有代理更新到下一个支持的次要版本。在引擎版本终止支持之前，Amazon MQ 会至少在 90 天内发出通知。 要查看版本支持日历和终止支持，请参阅以下内容： • 管理 Amazon MQ for ActiveMQ 引擎版本 • 管理 Amazon MQ for RabbitMQ 引擎版本 您还可以为代理启用自动次要版本升级，以便在维护时段内更新到下一个补丁版本。有关更多信息，请参阅升级 Amazon MQ 代理引擎版本
2024 年 5 月 9 日	适用于 RabbitMQ 的亚马逊 MQ 现在支持 RabbitMQ 3.12，这是一个次要版本。3.12.13 及更高版本的所有代理都使用经典队列版本 2 (CQv2)，3.12.13 及更高版本上的所有队列都表现为懒惰队列。 我们建议代理在 3.12.13 之前的版本上启用 CQv2 和延迟队列，或者升级到最新版本的 Amazon MQ for RabbitMQ。 有关此版本中的修复和特征的更多信息，请参阅以下内容： • RabbitMQ 服务器存储库上的 RabbitMQ 3.12 发行说明。 GitHub • 为您的 RabbitMQ 代理启用 Classic Queue v2 • 启用延迟队列 有关支持的 Amazon MQ for RabbitMQ 版本和代理升级的更多信息，请参阅 管理 Amazon MQ for RabbitMQ 引擎版本。

Date	文档更新
2024 年 3 月 4 日	适用于 RabbitMQ 的亚马逊 MQ 现在支持 RabbitMQ 3.11.28。 有关此版本中的修复和特征的更多信息，请参阅以下内容： • RabbitMQ 3.11.28 RabbitMQ 服务器存储库的发行说明 GitHub • RabbitMQ 更改日志 有关支持的 Amazon MQ for RabbitMQ 版本和代理升级的更多信息，请参阅 管理 Amazon MQ for RabbitMQ 引擎版本。
2024 年 1 月 19 日	适用于 RabbitMQ 的 Amazon MQ 不支持用户名 “访客”，并且将在您创建新经纪人时删除默认访客账户。Amazon MQ 还将定期删除任何由客户创建的名为 “访客” 的账户。
2023 年 12 月 15 日	Amazon MQ 现已在以色列（特拉维夫）区域推出。有关可用区域的信息，请参阅《AWS 一般参考》中的 AWS 区域和端点。
2023 年 12 月 11 日	Amazon MQ for RabbitMQ 现在支持 RabbitMQ 3.10.25。 有关此版本中的修复和特征的更多信息，请参阅以下内容： • RabbitMQ 3.10.25 关于 RabbitMQ 服务器存储库的发行说明 GitHub • RabbitMQ 更改日志 有关支持的 Amazon MQ for RabbitMQ 版本和代理升级的更多信息，请参阅 管理 Amazon MQ for RabbitMQ 引擎版本。
2023 年 10 月 26 日	Amazon MQ 发布了最新的 ActiveMQ 次要版本 5.15.16、5.16.7、5.17.6，并进行了一项关键更新。我们已经弃用了较旧的 ActiveMQ 次要版本，并将更新任何版本上的所有代理（版本 5.15 升级到 5.15.16、版本 5.16 升级到 5.16.7 以及版本 5.17 升级到 5.17.6）。 有关更新 ActiveMQ 代理的更多信息，请参阅管理 Amazon MQ for ActiveMQ 引擎版本。

Date	文档更新
2023 年 9 月 27 日	Amazon MQ for RabbitMQ 现在支持 RabbitMQ 3.11.20。 有关此版本中的修复和特征的更多信息，请参阅以下内容： • RabbitMQ 3.11.20 关于 RabbitMQ 服务器存储库的发行说明 GitHub • RabbitMQ 更改日志 有关支持的 Amazon MQ for RabbitMQ 版本和代理升级的更多信息，请参阅 管理 Amazon MQ for RabbitMQ 引擎版本。
2023 年 7 月 27 日	Amazon MQ for RabbitMQ 现在支持 RabbitMQ 3.11.16 有关此版本中的修复和特征的更多信息，请参阅以下内容： • RabbitMQ 3.11.16 关于 RabbitMQ 服务器存储库的发行说明 GitHub • RabbitMQ 更改日志 有关支持的 Amazon MQ for RabbitMQ 版本和代理升级的更多信息，请参阅 管理 Amazon MQ for RabbitMQ 引擎版本。
2023 年 7 月 27 日	Amazon MQ for RabbitMQ 现在支持为您的 RabbitMQ 代理创建和应用配置。 有关向代理添加配置的更多信息，请参阅RabbitMQ Broker Configurations。 有关此特征的更多信息，请参阅： • 操作员策略 • 操作员策略的变更

Date	文档更新
2023 年 6 月 23 日	Amazon MQ 现在支持 ActiveMQ 5.17.3，这是一个新的次要引擎版本。此版本支持 Amazon MQ 中新的跨区域数据复制（CRDR）特征。 有关更多信息，请参阅下列内容： • 要开始使用 CRDR，请参阅《开发人员指南》中的Amazon MQ for ActiveMQ 的跨区域数据复制。 • ActiveMQ 5.17.3 发布页面 • 管理 Amazon MQ for ActiveMQ 引擎版本 • 升级 Amazon MQ 代理引擎版本 • 使用 Spring XML 配置文件
2023 年 6 月 21 日	Amazon MQ for ActiveMQ 现在提供跨区域数据复制 (CRDR) 功能，允许将消息从主区域的主代理异步复制到副本区域的副本代理。 AWS 如果主区域中的主代理出现故障，则可以通过启动切换或失效转移，将辅助区域中的副本代理提升为主代理。 要开始使用 CRDR，请参阅《开发人员指南》中的Amazon MQ for ActiveMQ 的跨区域数据复制。
2023 年 5 月 18 日	Amazon MQ 现在已在以下区域推出： • 亚太地区（墨尔本） • 亚太地区（海得拉巴） • 欧洲（西班牙） • 欧洲（苏黎世） 有关可用区域的信息，请参阅《AWS 一般参考》中的 AWS 区域和端点。

Date	文档更新
2023 年 4 月 14 日	Amazon MQ for RabbitMQ 现在支持 RabbitMQ 版本 3.9.27。 有关此版本中的修复和特征的更多信息，请参阅以下内容： • RabbitMQ 3.9.27 关于 RabbitMQ 服务器存储库的发行说明 GitHub • RabbitMQ 更改日志 有关支持的 Amazon MQ for RabbitMQ 版本和代理升级的更多信息，请参阅 管理 Amazon MQ for RabbitMQ 引擎版本。
2023 年 4 月 14 日	Amazon MQ for RabbitMQ 现在支持 RabbitMQ 版本 3.10.20。 有关此版本中的修复和特征的更多信息，请参阅以下内容： • 关于 RabbitMQ 服务器存储库的 RabbitMQ 3.10.20 发行说明 GitHub • RabbitMQ 更改日志 有关支持的 Amazon MQ for RabbitMQ 版本和代理升级的更多信息，请参阅 管理 Amazon MQ for RabbitMQ 引擎版本。
2023 年 3 月 31 日	Amazon MQ for RabbitMQ 已禁用 RabbitMQ 引擎版本 3.10.17 Amazon MQ for RabbitMQ 团队和 RabbitMQ 的开源维护者已经发现，版本 3.10.17 上的 RabbitMQ 管理控制台存在问题。Amazon MQ 已撤回此版本。为了减轻此问题的影响，请在我们努力支持 RabbitMQ 的新补丁版本的同时，使用 3.10.20 版本创建新的代理。我们建议激活自动次要版本升级选项，以自动获取最新的错误修复、安全更新和性能增强。 有关可用的 Amazon MQ for RabbitMQ 版本的更多信息，请参阅 Amazon MQ for RabbitMQ 引擎版本。

Date	文档更新
2023 年 3 月 1 日	Amazon MQ for RabbitMQ 现在支持 RabbitMQ 版本 3.10.17。 有关此版本中的修复和特征的更多信息，请参阅以下内容： • RabbitMQ 3.10.17 关于 RabbitMQ 服务器存储库的发行说明 GitHub • RabbitMQ 更改日志 有关支持的 Amazon MQ for RabbitMQ 版本和代理升级的更多信息，请参阅 管理 Amazon MQ for RabbitMQ 引擎版本。
2023 年 2 月 21 日	适用于 RabbitMQ 的亚马逊 MQ 现已与 AWS Key Management Service (KMS) 集成，以提供服务器端加密。现在，您可以选择自己的客户托管 CMK，或者在 AWS KMS 账户中使用 AWS 托管 KMS 密钥。有关更多信息，请参阅 静态加密。 Amazon MQ 支持通过以下方式使用 AWS KMS 密钥。 • Amazon MQ 拥有的 KMS 密钥（默认值）– 密钥归 Amazon MQ 拥有和管理，且不在您的账户中。 • AWS 托管 KMS 密钥 — AWS 托管 KMS 密钥 (aws/mq) 是您账户中的 KMS 密钥，由 Amazon MQ 代表您创建、管理和使用。 • 选择现有的客户托管式 KMS 密钥 – 客户托管式 CMK 由您在 AWS Key Management Service (KMS) 中创建和管理。
2023 年 1 月 13 日	Amazon MQ for RabbitMQ 现在支持 RabbitMQ 版本 3.8.34。 有关此版本中的修复和特征的更多信息，请参阅以下内容： • RabbitMQ 3.8.34 关于 RabbitMQ 服务器存储库的发行说明 GitHub • RabbitMQ 更改日志 有关支持的 Amazon MQ for RabbitMQ 版本和代理升级的更多信息，请参阅 管理 Amazon MQ for RabbitMQ 引擎版本。

Date	文档更新
2022 年 12 月 15 日	Amazon MQ for RabbitMQ 现在支持 RabbitMQ 版本 3.9.24。 有关此版本中的修复和特征的更多信息，请参阅以下内容： • RabbitMQ 3.9.24 版本说明 RabbitMQ 服务器存储库 GitHub • RabbitMQ 更改日志 有关支持的 Amazon MQ for RabbitMQ 版本和代理升级的更多信息，请参阅 管理 Amazon MQ for RabbitMQ 引擎版本。
2022 年 12 月 13 日	Amazon MQ 现已在中东（阿联酋）区域推出。有关可用区域的信息，请参阅《AWS 一般参考》中的 AWS 区域和终端节点。
2022 年 11 月 14 日	Amazon MQ for RabbitMQ 现在支持 3.10，这是一个新的主要引擎版本。您现在可以对 RabbitMQ 队列启用 Classic Queues 版本 2 (CQv2)。不支持从 3.8 直接更新到 3.10。有关更多信息，请参阅下列内容： • RabbitMQ 3.10.10 发布说明 • RabbitMQ 更改日志 有关支持的 Amazon MQ for RabbitMQ 版本和代理升级的更多信息，请参阅 管理 Amazon MQ for RabbitMQ 引擎版本。
2022 年 11 月 9 日	Amazon MQ 现在支持 ActiveMQ 5.17.2，这是一个新的次要引擎版本。有关更多信息，请参阅下列内容： • ActiveMQ 5.17.2 发布说明 • 管理 Amazon MQ for ActiveMQ 引擎版本 • 升级 Amazon MQ 代理引擎版本 • 使用 Spring XML 配置文件

Date	文档更新
2022 年 8 月 17 日	Amazon MQ 现在支持 ActiveMQ 5.17.1，这是一个新的主要引擎版本。有关更多信息，请参阅下列内容： • ActiveMQ 5.17.1 发布说明 • 管理 Amazon MQ for ActiveMQ 引擎版本 • 升级 Amazon MQ 代理引擎版本 • 使用 Spring XML 配置文件
2022 年 7 月 14 日	Amazon MQ 现在支持 ActiveMQ 5.16.5，这是一个新的次要引擎版本。有关更多信息，请参阅下列内容： • ActiveMQ 5.16.5 发布说明 • 管理 Amazon MQ for ActiveMQ 引擎版本 • 使用 Spring XML 配置文件 • 升级 Amazon MQ 代理引擎版本
2022 年 5 月 4 日	Amazon MQ 在代理配置中添加了 `networkConnector` 元素的包容性语言。 • 创建和配置 Amazon MQ 代理网络
2022 年 4 月 25 日	Amazon MQ 这一版本添加了 `CRITICAL_ACTION_REQUIRED` 代理状态和 `ActionRequired` API 属性。当代理降级时，`CRITICAL_ACTION_REQUIRED` 会通知您。`ActionRequired` 为您提供一个代码，您可以使用该代码在《开发人员指南》中查找有关如何解决此问题的说明。 • the section called “故障排除：Amazon MQ 操作所需代码” • 《Amazon MQ API 参考》中的 `ActionRequired` 文档。
2022 年 4 月 20 日	Amazon MQ 现在支持 ActiveMQ 5.16.4，这是一个新的次要引擎版本。有关更多信息，请参阅下列内容： • ActiveMQ 5.16.4 发布说明 • 管理 Amazon MQ for ActiveMQ 引擎版本 • 使用 Spring XML 配置文件 • 升级 Amazon MQ 代理引擎版本

Date	文档更新
2022 年 3 月 1 日	Amazon MQ 现已在亚太地区（雅加达）区域推出。有关可用区域的信息，请参阅《AWS 一般参考》中的 AWS 区域和端点。
2022 年 2 月 25 日	Amazon MQ for RabbitMQ 现在支持 RabbitMQ version 3.8.27。 有关此版本中的修复和特征的更多信息，请参阅以下内容： • RabbitMQ 3.8.27 关于 RabbitMQ 服务器存储库的发行说明 GitHub • RabbitMQ 更改日志 有关支持的 Amazon MQ for RabbitMQ 版本和代理升级的更多信息，请参阅 管理 Amazon MQ for RabbitMQ 引擎版本。
2022 年 2 月 16 日	Amazon MQ 现已在非洲（开普敦）区域中推出。有关可用区域的信息，请参阅 AWS 一般参考中的 AWS 区域和终端节点。
2022 年 2 月 14 日	Amazon MQ for RabbitMQ 现在支持 RabbitMQ 版本 3.9.13。自动次要版本升级无法用于从 Rabbit 3.8 升级到 3.9。为此，请手动升级您的代理。 有关 RabbitMQ 3.9 中引入的新功能的更多信息，请参阅网站上 3.9.0 版本的发行说明页面。 GitHub **Note** 目前，Amazon MQ 不支持流，或 RabbitMQ 3.9 中推出的在 JSON 中使用结构化日志记录。 有关此版本中的修复和特征的更多信息，请参阅以下内容： • RabbitMQ 3.9.13 版本说明 RabbitMQ 服务器存储库 GitHub • RabbitMQ 更改日志 有关支持的 Amazon MQ for RabbitMQ 版本和代理升级的更多信息，请参阅 管理 Amazon MQ for RabbitMQ 引擎版本。

Date	文档更新
2022 年 2 月 7 日	Amazon MQ for RabbitMQ 推出新的代理指标，这些指标使您能够监控集群部署中的所有三个节点中的平均资源使用率。 有关更多信息，请参阅下列内容： • the section called “记录和监控 Amazon MQ for RabbitMQ 代理”
2022 年 1 月 18 日	Amazon MQ for RabbitMQ 现在支持 RabbitMQ 版本 3.8.26。 有关此版本中的修复和特征的更多信息，请参阅以下内容： • RabbitMQ 3.8.26 关于 RabbitMQ 服务器存储库的发行说明 GitHub • RabbitMQ 更改日志 有关支持的 Amazon MQ for RabbitMQ 版本和代理升级的更多信息，请参阅 管理 Amazon MQ for RabbitMQ 引擎版本。
2022 年 1 月 13 日	Amazon MQ 引入 RABBITMQ_MEMORY_ALARM 状态代码以在代理发出高内存警报并且处于运行不良状态时通知您。Amazon MQ 提供详细信息和建议来帮助您诊断、解决和防止高内存警报。有关更多信息，请参阅下列内容。 • the section called “ RABBITMQ_MEMORY_ALARM ”
2022 年 1 月 6 日	当您为 ActiveMQ 代理配置 Amazon MQ CloudWatch 日志时，Amazon MQ 支持在基于 IAM 资源的策略中aws:SourceArn 使用aws:SourceAccount 和全局条件上下文密钥来防止出现混淆的代理问题。有关更多信息，请参阅下列内容。 • the section called “跨服务混淆代理问题防范”
2021 年 12 月 20 日	Amazon MQ for ActiveMQ 引入一组新的指标，使您能够监控使用受支持的不同传输协议实现的最大连接数，并且其他新指标使您能够监控连接到代理网络中的代理的节点数。有关更多信息，请参阅下列内容。 • the section called “记录和监控 Amazon MQ for ActiveMQ 代理”

Date	文档更新
2021 年 11 月 16 日	Amazon MQ for RabbitMQ 现在支持 RabbitMQ 版本 3.8.23。 有关此版本中的修复和特征的更多信息，请参阅以下内容： • RabbitMQ 3.8.23 关于 RabbitMQ 服务器存储库的发行说明 GitHub • RabbitMQ 更改日志 有关支持的 Amazon MQ for RabbitMQ 版本和代理升级的更多信息，请参阅 管理 Amazon MQ for RabbitMQ 引擎版本。
2021 年 10 月 12 日	Amazon MQ 现在支持 ActiveMQ 5.16.3，这是一个新的次要引擎版本。有关更多信息，请参阅下列内容： • ActiveMQ 5.16.3 发布说明 • 管理 Amazon MQ for ActiveMQ 引擎版本 • 升级 Amazon MQ 代理引擎版本 • 使用 Spring XML 配置文件
2021 年 9 月 8 日	Amazon MQ for RabbitMQ 现在支持 RabbitMQ 版本 3.8.22。 此版本包含一个针对在以前支持的版本 RabbitMQ 3.8.17 中识别的队列问题的修复，这些队列使用每消息 TTL（存活时间）。我们建议您将现有代理升级到版本 3.8.22。 有关此版本中的修复和特征的更多信息，请参阅以下内容： • RabbitMQ 3.8.22 关于 RabbitMQ 服务器存储库的发行说明 GitHub • RabbitMQ 更改日志 有关支持的 Amazon MQ for RabbitMQ 版本和代理升级的更多信息，请参阅管理 Amazon MQ for RabbitMQ 引擎版本
2021 年 8 月 25 日	由于发现使用每条消息 (TTL) 的队列存在问题，适用于 RabbitMQ 的 Amazon MQ 暂时禁用了 RabbitMQ 引擎版本 3.8.17。 time-to-live 我们建议使用版本 3.8.11。

Date	文档更新
2021 年 7 月 29 日	Amazon MQ for RabbitMQ 现在支持 RabbitMQ 版本 3.8.17。有关此更新中包含的修补程序和特征的更多信息，请参阅以下内容： • RabbitMQ 3.8.17 版本说明 RabbitMQ 服务器存储库 GitHub • RabbitMQ 更改日志 • 管理 Amazon MQ for RabbitMQ 引擎版本
2021 年 7 月 16 日	现在，您可以使用 AWS Management Console、 AWS CLI或亚马逊 MQ API 调整亚马逊 MQ 经纪商的维护时段。要了解有关代理维护时段的更多信息，请参阅以下内容。 • 维护 Amazon MQ 代理
2021 年 7 月 6 日	Amazon MQ for RabbitMQ 推出了对一致性哈希交换器类型的支持。一致性哈希交换器根据通过消息的路由键计算的哈希值将消息路由到队列。有关更多信息，请参阅下列内容： • 一致性哈希交换器插件 • RabbitMQ 存储库上的 RabbitMQ 一致哈希交换类型 GitHub
2021 年 6 月 7 日	Amazon MQ 现在支持 ActiveMQ 5.16.2，这是一个新的主要引擎版本。有关更多信息，请参阅下列内容： • ActiveMQ 5.16.2 发布说明 • 管理 Amazon MQ for ActiveMQ 引擎版本 • 升级 Amazon MQ 代理引擎版本 • 使用 Spring XML 配置文件
2021 年 5 月 26 日	Amazon MQ for RabbitMQ 现已在中国（北京）和中国（宁夏）区域推出。有关可用区域的信息，请参阅AWS 区域和终端节点。

Date	文档更新
2021 年 5 月 18 日	Amazon MQ for RabbitMQ 实现了代理默认值。 首次创建代理时，Amazon MQ 会根据您选择的实例类型和部署模式创建一组代理策略和虚拟主机限制，以优化代理的性能。有关更多信息，请参阅下列内容： • Amazon MQ for RabbitMQ 代理默认值
2021 年 5 月 5 日	Amazon MQ 现在支持 ActiveMQ 5.15.15。有关更多信息，请参阅下列内容： • ActiveMQ 5.15.15 发布说明 • 管理 Amazon MQ for ActiveMQ 引擎版本 • 使用 Spring XML 配置文件
2021 年 5 月 5 日	Amazon MQ 开始跟踪 AWS 托管政策的变更。有关更多信息，请参阅下列内容： • the section called “AWS 托管策略”
2021 年 4 月 14 日	Amazon MQ 现已在中国（北京）和中国（宁夏）区域推出。有关可用区域的信息，请参阅AWS 区域和终端节点。
2021 年 4 月 7 日	Amazon MQ 现在支持 RabbitMQ 3.8.11。有关此更新中包含的修补程序和特征的更多信息，请参阅以下内容： • RabbitMQ 服务器存储库上的 RabbitMQ 3.8.11 发行说明 GitHub • RabbitMQ 更改日志 • 管理 Amazon MQ for RabbitMQ 引擎版本
2021 年 4 月 1 日	Amazon MQ 现已在亚太地区（大阪）区域推出。有关可用区域的信息，请参阅 Amazon MQ 区域和终端节点。

Date	文档更新
2020 年 12 月 21 日	Amazon MQ 现在支持 ActiveMQ 5.15.14。有关更多信息，请参阅下列内容： • ActiveMQ 5.15.14 发布说明 • 管理 Amazon MQ for ActiveMQ 引擎版本 • 使用 Spring XML 配置文件 • **Important** 由于此版本中存在已知的 Apache ActiveMQ 问题，ActiveMQ Web 控制台中的新 Pause Queue (暂停队列) 按钮不能用于 Amazon MQ for ActiveMQ 代理。有关此问题的信息，请参阅 AMQ-8104。
2020 年 11 月 4 日	Amazon MQ 现在支持 RabbitMQ，这是一个常用的开源消息代理。这使您 AWS 无需重写代码即可将现有的 RabbitMQ 消息代理迁移到。 Amazon MQ for RabbitMQ 可管理单个消息代理和集群消息代理，并处理预置基础设施、设置代理和更新软件等任务。 • Amazon MQ 支持 RabbitMQ 3.8.6。有关支持的引擎版本的更多信息，请参阅the section called “版本管理”。 • AWS 免费套餐包括长达 750 小时的单实例 `mq.t3.micro` 代理和高达每月 20GB 的存储空间，期限为一年。有关支持的实例类型的更多信息，请参阅Broker instance types。 • 借助 Amazon MQ for RabbitMQ，您可以在 RabbitMQ 客户端库支持的任何语言中，使用 AMQP 0-9-1 访问您的代理。有关受支持的协议和密码套件的更多信息，请参阅the section called “Amazon MQ for RabbitMQ 协议”。 • Amazon MQ for RabbitMQ 已在当前提供 Amazon MQ 的所有区域推出。要了解有关所有可用区域的更多信息，请参阅AWS 区域列表。 要开始使用 Amazon MQ，请创建代理，并将基于 JVM 的应用程序连接到您的 RabbitMQ 代理，请参阅the section called “创建并连接到 RabbitMQ 代理”。

Date	文档更新
2020 年 10 月 22 日	Amazon MQ 支持 ActiveMQ 5.15.13。有关更多信息，请参阅下列内容： • ActiveMQ 5.15.13 发布说明 • 管理 Amazon MQ for ActiveMQ 引擎版本 • 使用 Spring XML 配置文件
2020 年 9 月 30 日	Amazon MQ 现已在欧洲（米兰）区域推出。有关可用区域的信息，请参阅 Amazon MQ 区域和终端节点。
2020 年 7 月 27 日	您可以使用 Active Directory 或其他 LDAP 服务器中存储的凭据对 Amazon MQ 用户进行身份验证。您还可以添加、删除和修改 Amazon MQ 用户，并为主题和队列分配权限。有关更多信息，请参阅 将 LDAP 与 ActiveMQ 集成。
2020 年 7 月 17 日	Amazon MQ 现在支持 `mq.t3.micro` 实例类型。有关更多信息，请参阅 Broker instance types。
2020 年 6 月 30 日	Amazon MQ 支持 ActiveMQ 5.15.12。有关更多信息，请参阅下列内容： • ActiveMQ 5.15.12 发布说明 • 管理 Amazon MQ for ActiveMQ 引擎版本 • 使用 Spring XML 配置文件
2020 年 4 月 30 日	Amazon MQ 支持 `broker` 元素上的新子集合元素 `systemUsage` 。有关更多信息，请参阅 systemUsage。 Amazon MQ 还支持 `kahaDB` 子元素上的三个新属性。 • `journalDiskSyncStrategy=periodic` - 在 `journalDiskSyncInterval` 的情况下，执行磁盘同步的时间间隔 (毫秒)。 • `journalDiskSyncStrategy` - 配置磁盘同步策略。 • `preallocationStrategy` - 配置在需要新日志文件时，代理尝试预分配日志文件的方式。 有关更多信息，请参阅 属性。

Date	文档更新
2020 年 3 月 3 日	亚马逊 MQ 支持两个新指标 CloudWatch • `TempPercentUsage` – 非持久性消息使用的可用临时存储的百分比。 • `JobSchedulerStorePercentUsage` – 作业计划程序存储所使用的磁盘空间百分比。 有关更多信息，请参阅 Monitoring Amazon MQ using CloudWatch。
2020 年 2 月 4 日	Amazon MQ 已在亚太地区（香港）和中东（巴林）区域推出。有关可用区域的信息，请参阅AWS 区域和终端节点。
2020 年 1 月 22 日	Amazon MQ 支持 ActiveMQ 5.15.10。有关更多信息，请参阅下列内容： • ActiveMQ 5.15.10 发布说明 • 管理 Amazon MQ for ActiveMQ 引擎版本 • 使用 Spring XML 配置文件
2019 年 12 月 19 日	Amazon MQ 已在欧洲（斯德哥尔摩）和南美洲（圣保罗）区域推出。有关可用区域的信息，请参阅AWS 区域和终端节点。

Date	文档更新
2019 年 12 月 16 日	Amazon MQ 支持使用 Amazon Elastic Block Store（EBS）为代理存储创建吞吐量优化的代理，而不是使用默认的 Amazon Elastic File System（Amazon EFS）。要利用跨多个可用区的高持久性和复制功能，请使用 Amazon EFS。要利用低延迟和高吞吐量，请使用 Amazon EBS。 **Important** • 您只能将 Amazon EBS 与 `mq.m5` 代理实例类型系列配合使用。 • 尽管您可以更改代理实例类型，但在创建代理之后无法更改代理存储类型。 • Amazon EBS 在单个可用区内复制数据，但不支持 ActiveMQ 主动/备用部署模式。 有关更多信息，请参阅下列内容： • Storage • 选择正确的代理存储类型以实现最佳吞吐量 • 《Amazon MQ REST API 参考》中 `broker-instance-options` 资源的 `storageType` 属性 • Amazon MQ for ActiveMQ 指标 部分中的 `BurstBalance` 、`VolumeReadOps` 和 `VolumeWriteOps` 指标。
2019 年 10 月 18 日	有两个 Amazon CloudWatch 指标可用：`TotalEnqueueCount` 和`TotalDequeueCount` 。有关更多信息，请参阅 ActiveMQ 目标（队列和主题）指标。
2019 年 10 月 11 日	Amazon MQ 现已在美国商业区域中支持符合美国联邦信息处理标准 140-2（FIPS）的终端节点。 有关更多信息，请参阅下列内容： • 美国联邦信息处理标准 (FIPS) 140-2 • Amazon MQ 区域和终端节点

Date	文档更新
2019 年 9 月 30 日	Amazon MQ 现在提供了通过更改主机实例类型来扩展代理的功能。有关更多信息，请参阅 `UpdateBrokerInput` 的 `hostInstanceType` 属性，以及 `DescribeBrokerOutput` 的 `pendingHostInstanceType` 属性。
2019 年 8 月 30 日	现在，您可以在控制台中和使用 `UpdateBrokerInput` 更新与代理关联的安全组。
2019 年 7 月 22 日	Amazon MQ 与 AWS Key Management Service (KMS) 集成以提供服务器端加密。现在，您可以选择自己的客户托管 CMK，或者在 AWS KMS 账户中使用 AWS 托管 KMS 密钥。有关更多信息，请参阅 静态加密。 Amazon MQ 支持通过以下方式使用 AWS KMS 密钥。 • AWS 拥有的 KMS 密钥 — 密钥归亚马逊 MQ 所有，不在您的账户中。 • AWS 托管 KMS 密钥 — AWS 托管 KMS 密钥 (aws/mq) 是您账户中的 KMS 密钥，由亚马逊 MQ 代表您创建、管理和使用。 • 选择现有的客户托管式 CMK – 客户托管式 CMK 由您在 AWS Key Management Service （KMS）中创建和管理。
2019 年 6 月 19 日	Amazon MQ 已在欧洲（巴黎）和亚太地区（孟买）区域推出。有关可用区域的信息，请参阅AWS 区域和终端节点。
2019 年 6 月 12 日	Amazon MQ 已在加拿大（中部）区域推出。有关可用区域的信息，请参阅 AWS 区域和终端节点。
2019 年 6 月 3 日	有两个新的亚马逊 CloudWatch 指标可用：`EstablishedConnectionsCount` 和`InactiveDurableSubscribers` 。有关更多信息，请参阅下列内容： • Monitoring Amazon MQ using CloudWatch • Amazon MQ for ActiveMQ 指标

Date	文档更新
2019 年 5 月 10 日	新 `mq.t2.micro` 实例类型的数据存储限制为 20GB。有关更多信息，请参阅下列内容： • the section called “数据存储” • Broker instance types
2019 年 4 月 29 日	现在，您可以使用基于标签的策略和资源级权限。有关更多信息，请参阅下列内容： • Amazon MQ 如何与 IAM 协同工作 • Amazon MQ API 操作的资源级权限
2019 年 4 月 16 日	现在，您可以使用 REST API 检索有关代理引擎和代理实例选项的信息。有关更多信息，请参阅下列内容： • 代理实例选项 • 代理引擎类型
2019 年 4 月 8 日	Amazon MQ 支持 ActiveMQ 5.15.9。有关更多信息，请参阅下列内容： • ActiveMQ 5.15.9 发布说明 • 管理 Amazon MQ for ActiveMQ 引擎版本 • 使用 Spring XML 配置文件
2019 年 3 月 4 日	改进了有关为代理网络配置动态故障转移和重新平衡客户端的文档。通过配置 `transportConnectors` 以及 `networkConnectors` 配置选项启用动态故障转移。有关更多信息，请参阅下列内容： • 借助传输连接器进行的动态故障转移 • Amazon MQ 代理网络 • Amazon MQ Broker Configuration Parameters

Date	文档更新
2019 年 2 月 27 日	除了以下区域外，还在欧洲（伦敦）区域推出了 Amazon MQ： • 亚太地区（新加坡） • 美国东部（俄亥俄） • 美国东部（弗吉尼亚州北部） • 美国西部（加利福尼亚北部） • 美国西部（俄勒冈） • 亚太地区（东京） • 亚太地区（首尔） • 亚太地区（悉尼） • 欧洲地区（法兰克福） • 欧洲（爱尔兰）
2019 年 1 月 24 日	默认配置现在包含用于清除不活动目标的策略。
2019 年 1 月 17 日	Amazon MQ `mq.t2.micro` 实例类型目前仅支持每个线级协议 100 个连接。有关更多信息，请参阅Quotas in Amazon MQ。
2018 年 12 月 19 日	您可以在代理网络中配置一系列 Amazon MQ 代理。有关详细信息，请参阅以下章节： • Amazon MQ 代理网络 • Creating and Configuring a Network of Brokers • 正确配置您的代理网络 • `networkConnector` • `##ConnectionStart##`

2018 年 12 月 11 日	Amazon MQ 支持 ActiveMQ 5.15.8、5.15.6 和 5.15.0。
	• ActiveMQ 中已解决的错误和改进：
	  • ActiveMQ 5.15.8 发布说明
	  • ActiveMQ 5.15.7 发布说明

Date	文档更新
2018 年 12 月 5 日	AWS 支持资源标记，以帮助跟踪您的成本分配。您可以在创建资源时标记资源，也可以通过查看资源的详细信息来标记该资源。有关更多信息，请参阅标记资源。
2018 年 11 月 19 日	AWS 已扩大其 SOC 合规计划，将亚马逊 MQ 列为符合 SOC 标准的服务。
2018 年 10 月 15 日	• 每个用户的最大组数为 20。有关更多信息，请参阅 用户。 • 每个代理每个线级协议的最大连接数为 1000。有关更多信息，请参阅 代理。
2018 年 10 月 2 日	AWS 已扩大其 HIPAA 合规计划，将亚马逊 MQ 列为符合 HIPA A 资格的服务。

2018 年 9 月 27 日	除了 5.15.0 之外，Amazon MQ 还支持 ActiveMQ 5.15.6。有关更多信息，请参阅下列内容：
	• 编辑代理引擎版本、实例类型、 CloudWatch 日志和维护首选项
	• ActiveMQ 文档中已解决的错误和改进：
	  • ActiveMQ 5.15.6 发布说明
	  • ActiveMQ 5.15.5 发布说明
	  • ActiveMQ 5.15.4 发布说明
	  • ActiveMQ 5.15.3 发布说明
	  • ActiveMQ 5.15.2 发布说明
	  • ActiveMQ 5.15.1 发布说明
	• ActiveMQ 客户端 5.15.6

Date	文档更新

2018 年 8 月 31 日	• 可供使用的指标如下：
	  • `CurrentConnectionsCount`
	  • `TotalConsumerCount`
	  • `TotalProducerCount`
	  想要了解更多信息，请参阅Amazon MQ for ActiveMQ 指标部分。
	• 代理的 IP 地址显示在 Details (详细信息) 页面中。
	  Note
	  对于禁用公开可用性的代理，将显示内部 IP 地址。

2018 年 8 月 30 日	除以下区域外，还在亚太地区（新加坡）区域推出了 Amazon MQ： • 美国东部（俄亥俄） • 美国东部（弗吉尼亚州北部） • 美国西部（加利福尼亚北部） • 美国西部（俄勒冈） • 亚太地区（东京） • 亚太地区（首尔） • 亚太地区（悉尼） • 欧洲地区（法兰克福） • 欧洲（爱尔兰）
2018 年 7 月 30 日	您可以将 Amazon MQ 配置为向亚马逊日志发布一般日志和审核日志。CloudWatch 有关更多信息，请参阅 Configuring Amazon MQ to publish logs to Amazon CloudWatch Logs。

Date	文档更新
2018 年 7 月 25 日	除以下区域外，还在亚太地区（东京）和亚太地区（首尔）区域推出了 Amazon MQ ： • 美国东部（俄亥俄） • 美国东部（弗吉尼亚州北部） • 美国西部（加利福尼亚北部） • 美国西部（俄勒冈） • 亚太地区（悉尼） • 欧洲地区（法兰克福） • 欧洲（爱尔兰）
2018 年 7 月 19 日	您可以使用 AWS CloudTrail 记录亚马逊 MQ API 调用。有关更多信息，请参阅 Logging Amazon MQ API calls using CloudTrail。
2018 年 6 月 29 日	除了 `mq.t2.micro` 和 `mq.m4.large` 之外，以下代理实例类型可用于需要高吞吐量的常规开发、测试和生产工作负载： • `mq.m5.large` • `mq.m5.xlarge` • `mq.m5.2xlarge` • `mq.m5.4xlarge` 有关更多信息，请参阅 Broker instance types。
2018 年 6 月 27 日	除以下区域外，还在美国西部（加利福尼亚北部）区域推出了 Amazon MQ： • 美国东部（俄亥俄） • 美国东部（弗吉尼亚州北部） • 美国西部（俄勒冈州） • 亚太地区（悉尼） • 欧洲地区（法兰克福） • 欧洲（爱尔兰）

Date	文档更新

2018 年 6 月 14 日	• 您可以使用该`AWS::Amazon MQ::Broker` AWS CloudFormation 资源执行以下操作：
	  • 创建代理。
	  • 添加配置更改或修改指定代理的用户。
	  • 返回有关指定代理的信息。
	  • 删除指定代理。
	
	  Note
	  当您更改 Amazon MQ B roker ConfigurationId 或 Amazon MQ Broker 用户属性类型的任何属性时，该代理会立即重启。
	• 您可以使用该`AWS::Amazon MQ::Configuration` AWS CloudFormation 资源执行以下操作：
	  • 创建配置。
	  • 更新指定配置。
	  • 返回有关指定配置的信息。
	
	  Note
	  您可以使用 AWS CloudFormation 修改（但不能删除）Amazon MQ 配置。

2018 年 6 月 7 日	Amazon MQ 控制台支持德语、巴西葡萄牙语、西班牙语、意大利语和繁体中文。
2018 年 5 月 17 日	每个 代理的用户数限制为 250。有关更多信息，请参阅 用户。
2018 年 3 月 13 日	创建代理大约需要 15 分钟。有关更多信息，请参阅完成代理创建任务。

Date	文档更新
2018 年 3 月 1 日	• 您可以使用 ??? 属性为 Apache KahaDB 配置concurrentStoreAndDispatchQueues 并发存储和分派。 • 该CpuCreditBalance CloudWatch 指标适用于mq.t2.micro 代理实例类型。
2018 年 1 月 10 日	下列更改影响 Amazon MQ 控制台： • 在代理列表中，Creation (创建) 列默认情况下是隐藏的。如需自定义页面大小和列，请选择 。 • 在该*MyBroker*页面的 “连接” 部分，选择您的安全组名称或打开 EC2 控制台（而不是 VPC 控制台）。EC2 控制台可以对入站和出站规则进行更为直观的配置。想要了解更多信息，请参阅更新的启用入站连接部分。
2018 年 1 月 9 日	• 在 IAM 控制台上将 REST 操作 ID UpdateBroker 的权限正确列为 mq:UpdateBroker 。 • 从 IAM 控制台中删除了错误的 mq:DescribeEngine 权限。

Date	文档更新

2017 年 11 月 28 日	这是 Amazon MQ 和《Amazon MQ 开发人员指南》的初始版本。
	• Amazon MQ 已在以下区域推出：
	  • 美国东部（俄亥俄）
	  • 美国东部（弗吉尼亚州北部）
	  • 美国西部（俄勒冈州）
	  • 亚太地区（悉尼）
	  • 欧洲地区（法兰克福）
	  • 欧洲（爱尔兰）
	  对 `mq.t2.micro` 实例类型的使用受 CPU 积分和基准性能 的限制 – 能够突增 至基准性能水平以上（有关更多信息，请参阅 `CpuCreditBalance` 指标）。如果您的应用程序需要固定性能，请考虑使用 `mq.m5.large` 实例类型。
	• 您可以创建 `mq.m4.large` 和 `mq.t2.micro` 代理。
	  对 `mq.t2.micro` 实例类型的使用受 CPU 积分和基准性能 的限制 – 能够突增 至基准性能水平以上（有关更多信息，请参阅 `CpuCreditBalance` 指标）。如果您的应用程序需要固定性能，请考虑使用 `mq.m5.large` 实例类型。
	• 您可以使用 ActiveMQ 5.15.0 代理引擎。
	• 您还可以使用 Amazon MQ RES T API AWS 和软件开发工具包，以编程方式创建和管理经纪人。
	• 您可以访问您的代理，方法是使用 ActiveMQ 支持的任何编程语言并通过为以下协议明确启用 TLS：
	  • AMQP
	  • MQTT
	  • MQTT 结束了 WebSocket
	  • OpenWire
	  • STOMP
	  • 大吃一惊 WebSocket

Date	文档更新
	• 您可以使用各种 ActiveMQ 客户端连接到 ActiveMQ 代理。我们建议使用 ActiveMQ 客户端。有关更多信息，请参阅 Connecting a Java application to your broker。 • 您的代理可以发送和接收任何大小的消息。

Amazon MQ 文档历史记录

下表列出了对《Amazon MQ 开发人员指南》的更改。有关 Amazon MQ 特征发布和改进，请参阅 Amazon MQ 发布说明。

Date	文档更新
2022 年 8 月 22 日	已为 Amazon MQ for ActiveMQ 和 Amazon MQ for RabbitMQ 引擎创建了单独的父章节。这些父章节现在包含引擎详细信息、教程和最佳实践： • Working with Amazon MQ for ActiveMQ • Working with Amazon MQ for RabbitMQ
2022 年 1 月 13 日	添加了一个新的故障排查部分，其中列出了 Amazon MQ 在代理处于运行不良状态时返回的状态代码，以及与诊断和恢复代理相关的详细信息。 • the section called “故障排除：Amazon MQ 操作所需代码”
2021 年 11 月 8 日	添加了新教程，其中介绍了使用 Amazon MQ for RabbitMQ 代理设置 Python Pika 客户端。 • the section called “将 Python Pika 与 Amazon MQ for RabbitMQ 结合使用”
2021 年 10 月 8 日	已为 Amazon MQ for ActiveMQ 和 Amazon MQ for RabbitMQ 代理引擎添加以下问题排查主题： • 一些客户端无法连接 • the section called “在 Amazon MQ for RabbitMQ 中如何启用插件？” • the section called “我无法更改代理的 Amazon VPC 配置。”

Date	文档更新
2021 年 9 月 22 日	已添加为 Amazon MQ for ActiveMQ 代理的常见连接和授权问题进行问题排查的以下主题： • 重启后连接到代理 • Web 控制台上的 JSP 异常
2021 年 8 月 12 日	添加了以下部分，说明如何对使用 Amazon MQ 代理时的常见问题进行问题排查。 • 问题排查
2021 年 7 月 29 日	添加了以下部分来说明 Amazon MQ for RabbitMQ 版本管理，以及在支持时将 Amazon MQ 代理升级到新的次要和主要引擎版本。 • the section called “版本管理”
2021 年 7 月 21 日	添加了以下章节，介绍如何将 Amazon MQ 代理 AWS Lambda 作为事件源进行连接。 • Connect your Amazon MQ for ActiveMQ broker to Lambda • Connect your Amazon MQ for RabbitMQ broker to Lambda
2021 年 7 月 16 日	添加了以下章节，介绍亚马逊 MQ 代理维护时段，以及如何使用 AWS Management Console AWS CLI、或 Amazon MQ API 调整维护时段。 • the section called “维护代理”
2021 年 6 月 7 日	添加了以下部分来说明 Amazon MQ for ActiveMQ 版本管理，以及在支持时将 Amazon MQ 代理升级到新的次要和主要引擎版本。 • the section called “版本管理” • the section called “升级引擎版本”
2021 年 5 月 18 日	添加了以下部分来说明 Amazon MQ for RabbitMQ 代理默认设置 • the section called “代理默认设置”

Date	文档更新
2021 年 5 月 5 日	添加了以下部分，用于描述 Amazon MQ 的 AWS 托管策略以及这些政策的更新： • the section called “AWS 托管策略”
2021 年 2 月 16 日	添加了以下关于 Amazon MQ for RabbitMQ 的教程部分： • the section called “解决暂停队列同步的问题”

2020 年 11 月 4 日	• 添加了以下部分来记录 Amazon MQ for RabbitMQ 支持：
	  • the section called “创建并连接到 RabbitMQ 代理”
	  • the section called “RabbitMQ 教程”
	  • the section called “Amazon MQ for RabbitMQ 最佳实践”
	  • the section called “RabbitMQ 引擎”
	  • the section called “配置 Amazon MQ for RabbitMQ 日志”
	  • the section called “使用服务相关角色”
	• 对指南的现有章节和部分进行了其他修订，以准确记录 Amazon MQ for RabbitMQ 支持。

2019 年 12 月 16 日	• 增加了以下各节：
	  • Storage
	  • 选择正确的代理存储类型以实现最佳吞吐量
	• 修改了以下各节中的信息：
	  • 代理
	  • Broker instance types
	  • Amazon MQ 单实例代理
	  • 用于实现高可用性的 Amazon MQ 主动/备用代理
	  • Create an ActiveMQ broker
	  • Creating and configuring a broker

Date	文档更新

2019 年 7 月 19 日	在以下部分中修改和增加了有关加密管理方面的内容：
	• Amazon MQ 中的数据保护
	  • 静态加密
	  • 传输中加密
	• `EncryptionOptions`

2019 年 4 月 22 日	添加了以下有关基于标记的策略和资源级别权限的章节： • Amazon MQ 如何与 IAM 协同工作 • Amazon MQ API 操作的资源级权限
2019 年 3 月 4 日	改进了有关为代理网络配置动态故障转移和重新平衡客户端的文档。通过配置 `transportConnectors` 以及 `networkConnectors` 配置选项启用动态故障转移。 • 借助传输连接器进行的动态故障转移 • Amazon MQ 代理网络 • Amazon MQ Broker Configuration Parameters
2019 年 1 月 5 日	改进了有关一些每分钟指标的文档。有关更多信息，请参阅以下内容：ActiveMQ 目标（队列和主题）指标。

2018 年 12 月 19 日	• 增加了以下各节：
	  • Amazon MQ 代理网络
	  • Creating and Configuring a Network of Brokers
	  • 正确配置您的代理网络
	  • `networkConnector`
	  • `##ConnectionStart##`
	• 已将 `networkConnectors` 子集合元素添加到 Amazon MQ 配置中允许的元素、子集合元素及其子元素 部分。

2018 年 12 月 11 日	更新了文档，以反映 ActiveMQ 版本 5.15.8 的可用性。

Date	文档更新
2018 年 12 月 5 日	增加了为 资源添加标签部分。
2018 年 10 月 26 日	增加了通过恢复已准备 XA 事务避免缓慢重部分。
2018 年 10 月 15 日	已更新Quotas in Amazon MQ部分。
2018 年 10 月 1 日	更正了后续步骤部分中的信息。

2018 年 9 月 27 日	• 增加了编辑代理引擎版本、实例类型、 CloudWatch 日志和维护首选项部分。
	• 更新了以下部分：
	  • Create an ActiveMQ broker
	  • Configure Basic Broker Settings

2018 年 9 月 18 日	向 创建和管理 ActiveMQ 代理用户 部分中添加了以下注释：您不能独立于用户配置组。在向组标签中添加至少一个用户时，将创建组标签；在从组标签中删除所有用户时，将删除组标签。
2018 年 8 月 31 日	• 明确了主动/备用代理的术语。有关更多信息，请参阅 用于实现高可用性的 Amazon MQ 主动/备用代理。 • 简化了维护时段的术语。有关更多信息，请参阅 Amazon MQ 代理配置生命周期。 • 重新编写了Configure Additional Broker Settings部分。 • 更新了Amazon MQ for ActiveMQ 指标和Listing brokers and viewing broker details部分。
2018 年 8 月 15 日	更正了Create an ActiveMQ broker部分中的信息。
2018 年 8 月 13 日	增加了访问不可公开访问的代理 Web 控制台部分。

Date	文档更新
2018 年 8 月 2 日	• 增加了CloudWatch Logs 配置问题排查部分。 • 在本指南全文中增加了以下警告： **Important** 在以下示例代码中，生产者和使用者在单个线程中运行。对于生产系统（或测试代理实例故障转移），请确保您的创建者和使用者在单独的主机或线程上运行。
2018 年 8 月 1 日	更正了以下节中的信息： • 了解 CloudWatch Logs 中的日志记录的结构 • Connect a Java application to your broker
2018 年 7 月 31 日	• 将 3 分钟演示视频移动到了Getting Started with Amazon MQ节中。 • 将 3 分钟入门视频添加到了What is Amazon MQ?节中。
2018 年 7 月 30 日	• 增加了Configuring Amazon MQ to publish logs to Amazon CloudWatch Logs部分。 • 已更新Configure Additional Broker Settings部分。
2018 年 7 月 19 日	• 增加了Logging Amazon MQ API calls using CloudTrail部分。
2018 年 7 月 5 日	• 在`authorizationEntry` 部分增加了 始终配置授权映射 子元素交叉引用。 • 阐明了将 ActiveMQ 代理与 LDAP 集成部分中的信息。 • 阐明了API 限制部分中的信息。
2018 年 6 月 29 日	• 更新了 Broker instance types部分中的信息。 • 增加了选择正确的代理实例类型以实现最佳吞吐量部分。

Date	文档更新
2018 年 4 月 6 日	除了 HTML GitHub、PDF 和 Kindle 之外，亚马逊 MQ 开发者指南发行说明还以 RSS 提要的形式提供。 
2018 年 5 月 29 日	在Working Java Example一节中进行了以下更改： • 添加了一个 STOMP+WSS Java 示例。STOMP+WSS 示例 Java 代码可连接到代理、创建队列并发送和接收消息。 • 改进了 MQTT Java 示例。 • 改进了 OpenWire Java 示例。
2018 年 5 月 24 日	更正了Working Java Example节中的 MQTT Java 示例中的线级协议端点端口。
2018 年 5 月 22 日	更正了所有 Java 依赖项部分中的信息。
2018 年 5 月 17 日	更正了用户部分中的信息。
2018 年 5 月 15 日	更正了确保有效的 Amazon MQ 性能部分中的信息。
2018 年 5 月 8 日	• 已将 Amazon MQ REST API 权限参考放入其自己的部分中。 • 使用示例自定义 IAM 策略创建了要创建 Amazon MQ 代理所需的 IAM 权限部分。
2018 年 5 月 7 日	• 本指南中阐明了代理维护时段为 2 小时。有关更多信息，请参阅 Amazon MQ 代理配置生命周期。 • 添加了有关 `ec2:CreateNetworkInterface` 和 `ec2:CreateNetworkInterfacePermission` 权限对于创建代理是必需的原因说明。有关更多信息，请参阅 Amazon MQ 的 API 身份验证和授权。

Date	文档更新
2018 年 5 月 1 日	在以下部分中阐明了有关主动/备用代理的维护时段的信息： • 用于实现高可用性的 Amazon MQ 主动/备用代理 • Creating and configuring a broker • Creating and applying broker configurations
2018 年 4 月 27 日	重新编写了以下部分并优化了示例 Java 代码以匹配仅对创建者而非使用者使用连接池的建议： • 始终使用连接池 • 创建消息创建者并发送消息 • 创建消息使用者并接收消息 • AmazonMQExample.java
2018 年 4 月 26 日	在 Working Java Example 部分中添加了 MQTT Java 示例。MQTT 示例 Java 代码可连接到代理、创建主题并发送和接收消息。
2018 年 4 月 4 日	将“与 Amazon MQ 通信”部分重命名为连接到 Amazon MQ。
2018 年 4 月 3 日	阐明并更正了 对具有慢速使用者的队列禁用并发存储和分派部分中的信息。
2018 年 4 月 2 日	将“Amazon MQ 中队列的并发存储和分派”部分移动到了对具有慢速使用者的队列禁用并发存储和分派部分。

2018 年 3 月 27 日	• 在节中，将 re:Invent 发布视频替换为 3 分钟演示视频What is Amazon MQ?。
	• 重新组织了以下部分的结构：
	  • Broker Architecture
	  • Amazon MQ 的工作原理。
	• 将Amazon MQ 代理配置生命周期移到了Broker Architecture部分下面。

Date	文档更新
2018 年 3 月 22 日	本指南澄清了以下内容：Amazon MQ 使用其管理和安全存储的加密密钥对静态消息和传输中的消息进行加密。有关更多信息，请参见AWS Encryption SDK 开发人员指南。
2018 年 3 月 19 日	本指南澄清了以下内容：主动/备用代理由两个不同可用区中的两个代理组成，配置为冗余对。这些代理与您的应用程序以及 Amazon EFS 进行同步通信。
2018 年 3 月 15 日	• 已重构Amazon MQ Basic elements 部分。
2018 年 3 月 12 日	• 阐明并更正了Amazon MQ 的安全最佳实践和连接到 Amazon MQ部分中的信息。 • 增加了对具有慢速使用者的队列禁用并发存储和分派部分。 • 将 admonitions 组合到配置高级代理设置部分的前言中。
2018 年 3 月 9 日	• 阐明并更正了 始终配置授权映射部分中的信息。 • 增加了authorizationEntry部分并更新了kahaDB部分。
2018 年 3 月 8 日	• 增加了始终配置授权映射部分。 • 向Monitoring Amazon MQ using CloudWatch部分中添加了有关代理后缀的注释。
2018 年 3 月 6 日	在本指南全文中添加了以下说明： **Note** 对 `mq.t2.micro` 实例类型的使用受 CPU 积分和基准性能 的限制 – 能够突增 至基准性能水平以上（有关更多信息，请参阅 `CpuCreditBalance` 指标）。如果您的应用程序需要固定性能，请考虑使用 `mq.m5.large` 实例类型。

Date	文档更新
2018 年 3 月 1 日	• 将 `CpuCreditBalance` 指标添加到了Amazon MQ for ActiveMQ 指标节中。 • 增加了Amazon MQ 子元素属性部分。 • 在the section called “允许的元素”部分中添加了从元素指向其属性和子集合元素的链接。 • 对中的 AWS 词汇表进行了更正 GitHub。
2018 年 2 月 28 日	更正了中的图像显示 GitHub。
2018 年 2 月 27 日	除了 HTML、PDF 和 Kindle 之外，亚马逊 MQ 开发者指南也可在以下网址获得。 GitHub要留下反馈，请选择右上角的 GitHub 图标。 
2018 年 2 月 26 日	• 使所有示例和图表中的区域保持一致。 • 优化了指向 AWS 控制台和产品网页的链接。
2018 年 2 月 22 日	阐明并更正了以下部分中的信息： • 首选不可公开访问的代理 • 始终使用故障转移传输连接到多个代理终端节点 • Amazon MQ 的 API 身份验证和授权 • 将 ActiveMQ 代理与 LDAP 集成
2018 年 2 月 21 日	更正了以下部分中的 Java 代码： • Working Java Example • Connect a Java application to your broker • 始终使用连接池
2018 年 2 月 20 日	阐明并更正了Amazon MQ 中的安全性和“最佳实践”部分中的信息。

Date	文档更新
2018 年 2 月 19 日	• 更正了始终使用连接池部分中的 Java 代码。 • 重新设计并扩展了“最佳实践”部分和Amazon MQ 中的安全性部分。

2018 年 2 月 16 日	• 增加了Amazon MQ 的安全最佳实践部分。
	• 已更新连接到 Amazon MQ部分。
	• 更正了以下部分中的 Java 代码：
	  • Getting Started with Amazon MQ
	  • AmazonMQExample.java

2018 年 2 月 15 日	• 重新设计并扩展了“最佳实践”部分。
	• 更新了以下部分：
	  • 如何开始使用 Amazon MQ？
	  • 后续步骤 (入门)
	  • Related resources

2018 年 2 月 14 日	更新了以下部分： • Quotas in Amazon MQ • API 限制 • Amazon MQ 中的安全性
2018 年 2 月 13 日	• 已更新Related resources部分。 • 已更新Quotas in Amazon MQ部分。 • 增加了我们希望听到您的意见和建议部分。
2018 年 1 月 25 日	• 修复了添加 Java 依赖项部分的Working Java Example子部分中的错误。 • 在 IAM 控制台上将 REST 操作 ID `RebootBroker` 的权限正确列为 `mq:RebootBroker` 。
2018 年 1 月 24 日	• 增加了永远不要修改或删除 Amazon MQ 弹性网络接口部分。 • 更新了本指南中的所有图表。 • 在本指南中添加了指向 Amazon MQ REST API 参考 的链接，并在 Amazon MQ 的 API 身份验证和授权 部分中添加了指向特定 REST API 的链接。

Date	文档更新
2018 年 1 月 19 日	更新了 Amazon MQ for ActiveMQ 资源部分中的信息。
2018 年 1 月 18 日	阐明并更正了 Quotas in Amazon MQ部分中的信息。
2018 年 1 月 17 日	恢复了首选虚拟目标而非持久订阅的建议，并改进了说明。
2018 年 1 月 11 日	• 除了 HTML 和 PDF 格式外，还提供 Kindle 格式的《Amazon MQ 开发人员指南》https://docs.aws.amazon.com/amazon-mq/latest/developer-guide/amazon-mq-dg.pdf。 • 阐明并更正了Amazon MQ 的 API 身份验证和授权和第 2 步：创建用户并获取您的 AWS 证书部分中的信息。
2018 年 1 月 3 日	在 Amazon MQ 的 API 身份验证和授权 部分中增加了 `DescribeConfigurationRevision` 。
2017 年 12 月 15 日	从“最佳实践”部分中删除了针对持久订阅的建议。
2017 年 12 月 8 日	• 在 启用入站连接 和 Connecting a Java application to your broker 部分中增加了 Working Java Example 先决条件。 • 在本指南中添加了以下说明：当前，您无法删除配置。
2017 年 12 月 7 日	• 改进了 AmazonMQExample.java 中的代码。 • 增加了Amazon MQ 的 API 身份验证和授权部分。

2017 年 12 月 5 日	• 阐明并更正了 Monitoring Amazon MQ using CloudWatch 部分中的信息：
	  • 改进了指标描述。
	  • 增加了 Amazon MQ for ActiveMQ 指标 和 ActiveMQ 代理指标的维度 小节。
	• 在What is Amazon MQ?部分中添加了“Amazon MQ 简介”视频。

Date	文档更新
2017 年 12 月 4 日	• 在 数据存储 部分中阐明了以下信息：每个代理的存储容量为 200 GB。 • 在先决条件部分中添加了 Working Java Example。（实例需要安装 `activemq-client.jar` 和 `activemq-pool.jar` 软件包才能工作。有关更多信息，请参阅 Connecting a Java application to your broker）。
2017 年 12 月 1 日	• 更新和改进了所有教程中的屏幕截图。 • 本指南阐明了以下内容：对配置修订版或 ActiveMQ 用户进行更改不会立即应用更改。要应用更改，必须等待下一维护时段或者重启代理。有关更多信息，请参阅 Amazon MQ 代理配置生命周期。

AWS 术语表

有关最新的 AWS 术语，请参阅《AWS 词汇表参考》中的 AWS 词汇表。

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。