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 Apache ActiveMQ Classic 和 RabbitMQ 的托管式消息代理服务，可管理消息代理的设置、运行和维护。您可以使用行业标准消息传递协议创建新的 Amazon MQ 代理，或将现有消息代理迁移到 Amazon MQ，而无需重写消息传递代码。

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。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。您可以使用 Amazon MQ 代理在大型云原生应用程序和组件之间进行通信。

主题

Amazon MQ 功能

托管式维护和版本升级

Amazon MQ 会在您计划的维护窗口期间对消息代理执行维护和版本升级。

使用 CloudWatch 监控代理

Amazon MQ 与 Amazon CloudWatch 集成，因此您可以查看和分析代理和队列的指标。您可以通过 Amazon MQ 控制台、CloudWatch 控制台、命令行和 API 查看和分析指标。每分钟自动收集指标并推送至 CloudWatch。

安全性。

Amazon MQ 可对静态和传输中的消息提供加密。与代理的连接使用 SSL，访问权限可限制在 Amazon VPC 内的私有端点。此外，您可以使用 AWS Identity and Access Management (IAM) 来控制 IAM 用户和组在特定 Amazon MQ 代理上的操作。

RabbitMQ on Amazon MQ 的仲裁队列

仲裁队列是一种复制队列类型，由领导节点（主副本）和跟随节点（其他副本）组成。每个节点都位于不同的可用区中，因此，如果一个节点暂时不可用，信息将通过另一个可用区新选出的领导副本继续传递。仲裁队列可用于处理毒丸消息，当消息失败并被多次重新排队时，就会出现毒丸消息。

ActiveMQ on Amazon MQ 的跨区域数据复制

跨区域数据复制（CRDR）允许从主 AWS 区域的主代理向副本区域的副本代理进行异步消息复制。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。

如何开始使用 Amazon MQ？

要开始使用 ActiveMQ on Amazon MQ，请查看以下文档：

- 入门：创建并连接 ActiveMQ 代理
- the section called “部署代理”
- ActiveMQ 教程
- the section called “Amazon MQ for ActiveMQ 最佳实践”

要开始使用 RabbitMQ on Amazon MQ，请查看以下文档：

- 入门：创建并连接 RabbitMQ 代理
- the section called “部署 RabbitMQ 代理”
- the section called “RabbitMQ 教程”
- the section called “Amazon MQ for RabbitMQ 最佳实践”

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

如需了解 Amazon MQ AWS CLI 命令，请参阅 AWS CLI 命令参考 中的 Amazon MQ。

如何向 Amazon MQ 提供反馈？

我们欢迎并鼓励您对文档提供反馈。您可以使用右侧的大拇指向上和向下图标提交反馈，也可以使用下面链接的“提供反馈”表单。

要联系 Amazon MQ 团队，请使用 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

注册获取 AWS 账户

首先 AWS，你需要一个 AWS 账户。有关创建的信息 AWS 账户，请参阅《AWS 账户管理 参考指南》 AWS 账户中的入门指南。

准备好使用示例代码吧

以下教程展示了如何使用与亚马逊 MQ 经纪人合作， AWS 管理控制台 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 应用程序连接到适用于 ActiveMQ 的 Amazon MQ 代理，或者使用 RabbitMQ Jav a 客户端库将应用程序连接到适用于 RabbitMQ 的 Amazon MQ 代理。 JVM-based

入门：创建并连接 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`) 和大小 (`large`, `medium`) 的组合描述称为代理实例类型（例如，`mq.m5.large`）。有关更多信息，请参阅 什么是 Amazon MQ for ActiveMQ 代理？。

创建 ActiveMQ 代理

第一个也是最常见的 Amazon MQ 任务是创建代理。以下示例演示如何使用AWS 管理控制台创建基本代理。

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 Amazon MQ for ActiveMQ 代理的部署选项。

      - 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 选项 1：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 选项 2：亚马逊 MQ active/standby 代理以实现高可用性。

   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。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。

> **Note**
>
> 在其他设置部分，您还可以配置以下内容：
>
> - 配置
> - CloudWatch Logs
> - 私有访问
> - 代理维护窗口

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 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。

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 ***MyBroker*** 页面的连接部分，记下代理的 ActiveMQ Web 控制台 URL，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
   ```

入门：创建并连接 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) 和大小 (large, medium) 的组合描述称为代理实例类型（例如，mq.m5.large）。有关更多信息，请参阅 什么是 Amazon MQ for RabbitMQ 代理？。

创建 RabbitMQ 代理

第一个也是最常见的 Amazon MQ 任务是创建代理。以下示例演示如何使用AWS 管理控制台创建基本代理。

创建 Amazon MQ for RabbitMQ 代理时，请遵循 RabbitMQ 代理设置最佳实践以最大化代理性能并优化消息吞吐效率。

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 选项 1：Amazon MQ for RabbitMQ 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 选项 2：Amazon MQ for RabbitMQ 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。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. 选择代理实例类型（例如，mq.m7g.large）。有关更多信息，请参阅 Broker instance types。

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 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。

   > **Note**
   >
   > 在其他设置部分，您还可以配置以下内容：
   >
   > - 配置
   > - CloudWatch Logs
   > - 私有访问
   > - 代理维护窗口

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*** 页面的连接部分，记下代理的 RabbitMQ Web 控制台 URL，例如：

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

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.on.aws:5671
```

管理 Amazon MQ 代理

创建代理后，您可以管理和维护 Amazon MQ 代理的不同组件。

主题

连接到 Amazon MQ

您可以使用服务端点和代理端点从其他 AWS 服务连接到 Amazon MQ。

服务端点

以下连接方法用于 Amazon MQ 服务 API：

域	连接方法
`mq.region.amazonaws.com`	IPv4
`mq.region.api.aws`	双栈（IPv4 和 IPv6）
`mq-fips.region.amazonaws.com`	仅 IPv4 的 FIPS
`mq-fips.region.api.aws`	双栈 FIPS

代理端点

以下连接方法用于 Amazon MQ 代理：

域	连接方法
brokerId.mq.*region*.amazonaws.com	IPv4
brokerId.mq.*region*.on.aws	双栈（IPv4 和 IPv6） **Note** Amazon MQ for ActiveMQ 代理不支持双栈。

使用双栈（IPv4 和 IPv6）端点连接到 Amazon MQ

同时支持 IPv4 和 IPv6 流量的双堆栈端点。当您向双栈端点发出请求时，端点 URL 会解析为 IPv4 或 IPv6 地址。有关双栈和 FIPS 端点的更多信息，请参阅《SDK 参考指南》。

Amazon MQ 支持区域双栈端点，这意味着您必须将 AWS 区域指定为端点名称的一部分。双栈端点名称使用以下命名约定：mq.*region*.api.aws。例如，eu-west-1 区域的双堆栈端点名称是 mq.eu-west-1.api.aws。

有关 Amazon MQ 端点的完整列表，请参阅《AWS 通用参考》。

使用 AWS PrivateLink 连接到 Amazon MQ

支持 IPv4 和 IPv6 的 Amazon MQ API 的 AWS PrivateLink 端点在虚拟私有云（VPC）和 Amazon MQ API 之间提供私有连接，不会将您的流量暴露到公共互联网。

Note

对 PrivateLink 的支持仅适用于 Amazon MQ API 端点，不适用于代理端点。有关私有连接到代理端点的更多信息，请参阅Configuring a private Amazon MQ broker。

要使用 PrivateLink 访问 Amazon MQ API，您必须首先在要连接的特定 VPC 中创建一个接口 VPC 端点。创建 VPC 端点时，使用服务名称 com.amazonaws.*region*.mq 或 com.amazonaws.*region*.mq-fips 用于 FIPS 端点。

当您使用 AWS CLI 或 SDK 调用 Amazon MQ 时，必须指定端点 URL 以使用双栈域名：mq.*region*.api.aws 或 mq-fips.*region*.api.aws。Amazon MQ 的 PrivateLink 不支持以 amazonaws.com 结尾的默认域名。有关更多信息，请参阅 SDK 参考指南中的双栈和 FIPS 端点。

以下 CLI 示例展示了如何通过 Amazon MQ VPC 端点调用亚太地区（悉尼）区域的 describe-broker-engine-type。

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

有关在 CLI 中配置端点的其他方法，请参阅在 AWS CLI 中使用端点

您还可以使用 VPC 端点策略来确定用户对 VPC 端点的访问权限。有关更多信息，请参阅使用端点策略控制对 VPC 端点的访问。

Amazon MQ 经纪人的身份验证和授权

Amazon MQ 提供多种身份验证和授权方法，可根据贵组织的要求保护您的消息基础设施。

适用于 RabbitMQ 的亚马逊 MQ 的身份验证和授权

Amazon MQ for RabbitMQ 支持以下认证与授权方法：

简单认证与授权

在此方法中，代理用户存储在 RabbitMQ 代理内部，并通过 Web 控制台或管理 API 进行管理。虚拟主机、交换机、队列和主题的权限直接在 RabbitMQ 中配置。这是默认方法。有关更多信息，请参阅简单身份验证和授权。

OAuth 2.0 认证与授权

在此方法中，代理用户及其权限由外部 OAuth 2.0 身份提供者（IdP）管理。虚拟主机、交换机、队列和主题的用户认证和资源权限通过 OAuth 2.0 提供程序的范围系统进行集中管理。这简化了用户管理，并实现了与现有身份系统的集成。有关更多信息，请参阅 OAuth 2.0 身份验证和授权。

IAM 身份验证和授权

在此方法中，代理用户通过 IAM 出站联合使用 AWS IAM 凭证进行身份验证。IAM 证书用于从 AWS 安全令牌服务 (STS) 获取 JWT 令牌，而这些 JWT 令牌则用作 OAuth 2.0 令牌进行身份验证。此方法利用了亚马逊 MQ 中对 RabbitMQ 的现有 OAuth 2.0 支持，RabbitMQ 充当 OAuth 2.0 身份提供商。 AWS 用户身份验证由 AWS IAM 处理，而虚拟主机、交易所、队列和主题的资源权限则通过 RabbitMQ 中配置的 IAM 策略和范围别名进行管理。有关更多信息，请参阅 IAM 身份验证和授权。

LDAP 身份验证和授权

在这种方法中，代理用户及其权限由外部 LDAP 目录服务管理。用户身份验证和资源权限通过 LDAP 服务器集中管理，允许用户使用其现有的目录服务凭据访问 RabbitMQ。有关更多信息，请参阅 LDAP 身份验证和授权。

HTTP 身份验证和授权

在这种方法中，代理用户及其权限由外部 HTTP 服务器管理。用户身份验证和资源权限通过 HTTP 服务器集中管理，允许用户使用自己的身份验证和授权提供程序访问 RabbitMQ。有关此方法的更多信息，请参阅 HTTP 身份验证和授权。

SSL 证书身份验证

亚马逊 MQ 支持 RabbitMQ 经纪商的双向 TLS (mTLS)。SSL 身份验证插件使用来自 mTLS 连接的客户端证书对用户进行身份验证。在这种方法中，使用 X.509 客户端证书而不是用户名和密码凭证对经纪人用户进行身份验证。根据受信任的证书颁发机构 (CA) 对客户端的证书进行验证，用户名是从证书的字段（例如公用名 (CN) 或主题备用名称 (SAN)）中提取的。此方法无需通过网络传输凭据即可提供强身份验证。有关更多信息，请参阅 SSL 证书身份验证。

> **Note**
>
> RabbitMQ 支持同时使用多种身份验证和授权方法。例如，您可以同时启用 OAuth 2.0 和简单（内部）身份验证。有关更多信息，请参阅 OAuth 2.0 教程中关于同时启用 OAuth 2.0 和简单（内部）身份验证的部分以及 Rabb itMQ 访问控制文档。

适用于 ActiveMQ 的亚马逊 MQ 的身份验证和授权

适用于 ActiveMQ 的亚马逊 MQ 支持以下身份验证和授权方法：

简单认证与授权

在这种方法中，代理用户是通过 Amazon MQ 控制台或 API 创建和管理的。可以为用户配置访问队列、主题和 ActiveMQ Web 控制台的特定权限。有关此方法的更多信息，请参阅创建 ActiveMQ 代理用户。

LDAP 身份验证和授权

在这种方法中，代理用户通过存储在您的 LDAP 服务器中的凭据进行身份验证。您可以通过 LDAP 服务器添加、删除和修改用户以及为主题和队列分配权限，从而提供集中式身份验证和授权。有关此方法的更多信息，请参阅将 ActiveMQ 代理与 LDAP 集成。

升级 Amazon MQ 代理引擎版本

Amazon MQ 会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将版本号整理为 `X.Y.Z`。在 Amazon MQ 实现中，`X` 表示主要版本，`Y` 表示次要版本，`Z` 表示补丁版本号。Amazon MQ 支持两种类型的升级：

- 主要版本升级 – 当主要引擎版本号更改时发生。例如，从 RabbitMQ 版本 3. 13 升级到版本 4 .2 被视为主要版本升级。
- 次要版本升级 – 仅在次要引擎版本号更改时发生。例如，从版本 3 升级。 11 到版本 3。 12 被视为次要版本升级。

您可以随时手动将代理升级到下一个受支持的主要版本或次要版本。Amazon MQ 在计划维护时段内为所有代理管理升级到支持的最新补丁版本。手动和自动版本升级都是在定期维护时段内进行的，或者是在您重启代理之后。当前次要版本的支持终止时，Amazon MQ 会将代理升级到下一个次要版本。

手动升级引擎版本

您可以使用 AWS 管理控制台、或 Amazon MQ API 升级代理的引擎版本。 AWS CLI

AWS 管理控制台

要升级代理的引擎版本，请使用 AWS 管理控制台

1. 在代理详细信息页上，选择 Edit (编辑)。

2. 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。
3. 滚动到页面底部并选择 Schedule modifications (计划修改)。
4. 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**
   >
   > 单实例代理在重启期间处于离线状态。对于集群代理，在代理重启期间，每次只有一个节点宕机。

5. 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 reboo t-broker CLI 命令重启您的代理。

   ```
   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**
>
> 单实例代理在重启期间处于离线状态。对于集群代理，在代理重启期间，每次只有一个节点宕机。

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**
>
> 单实例代理在重启期间处于离线状态。对于集群代理，在代理重启期间，每次只有一个节点宕机。

升级 Amazon MQ 代理实例类型

> **Important**
>
> `mq.m7g.x` 实例仅适用于 Amazon MQ for RabbitMQ 代理。Amazon MQ for ActiveMQ 代理仅使用 `mq.m5.x` 实例。

代理实例类 (`m7g`) 和大小 (`large`) 的组合描述称为代理实例类型（例如，`mq.m7g.large`）。选择实例类型时，必须考虑会影响代理性能的因素：

- 客户端和队列的数量
- 发送的消息量
- 保存在内存中的消息
- 冗余消息

建议仅在测试应用程序性能时使用较小的代理实例类型 (`mq.m7g.medium`)。对于生产级别的客户端和队列、高吞吐量、内存中的消息和冗余消息，我们建议使用较大的代理实例类型（`mq.m7g.large` 及以上）。

如果您遇到性能问题，或者从测试环境迁移到生产环境，我们建议升级到更大的实例类型（即从 `micro` 到 `large`）。要升级您的实例类型，您可以使用 AWS 管理控制台、AWS CLI 或 Amazon MQ API。

AWS 管理控制台

要使用 AWS 管理控制台 升级到更大的实例类型，请执行以下操作：

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. 在规格下，对于代理实例类型，从下拉列表中选择新的实例类型。
5. 在页面底部，选择计划修改。
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。
   - 选择下次重启后，如果您希望 Amazon MQ 在下一个计划维护窗口中完成升级。
   - 选择立即，如果您希望立即重启代理并升级实例类型。

   > **Important**
   >
   > 单实例代理在重启期间处于离线状态。对于集群代理，在代理重启期间，每次只有一个节点宕机。

7. 选择 Apply (应用) 以完成应用更改。

AWS CLI

要使用 AWS CLI 升级代理的实例类型

1. 使用 modify-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。
   - `--host-instance-type` – 代理引擎要升级到的版本号。

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

2. （可选）如果您想立即升级实例类型，请使用 reboot-broker CLI 命令重启代理。

   ```
   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**
>
> 单实例代理在重启期间处于离线状态。对于集群代理，在代理重启期间，每次只有一个节点宕机。

Amazon MQ API

要使用 Amazon MQ API 升级代理的实例类型

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 Amazon MQ 端点的更多信息，请参阅 AWS 一般参考 中的 Amazon MQ 端点和配额。

   ```
   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
   ```

   在请求有效载荷中使用 `host-instance-type` 指定代理要升级到的实例类型。

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

2. （可选）如果您想立即升级引擎版本，请使用 RebootBroker API 操作重新启动您的代理。`broker-id` 已指定为路径参数。

   ```
   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**
>
> 单实例代理在重启期间处于离线状态。对于集群代理，在代理重启期间，每次只有一个节点宕机。

Amazon MQ for ActiveMQ 存储类型

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	Persistence	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 • 位置数据更新 • 频繁更改的数据	5000	无
Amazon EBS	持续的	• 大量文本	500	单个可用区 (AZ) 内的多个副本

存储类型	Persistence	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
		• 订单处理		
Amazon EFS	持续的	金融交易	80	跨多个副本 AZs

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 的设计具有很高的耐久性，可以跨多个组件进行复制， AZs 以防止由于任何单个组件的故障或影响可用区可用性的问题而导致的数据丢失。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。

配置私有 Amazon MQ 代理

私有代理不具备公共访问权限，无法从您的 VPC 外部访问。在配置私有代理之前，请查看有关 VPCs 子网和安全组的以下信息：

- VPCs
  - 代理的子网和安全组必须位于同一 VPC 中。
  - 使用私有代理时，您可能会看到未在 VPC 中配置的 IP 地址。这些 IP 地址来自 Amazon MQ 基础设施，无需任何操作。
- 子网
  - 如果子网位于共享 VPC 内，则该 VPC 必须由创建代理的同一账户拥有。
  - 如果未提供子网，将使用默认 VPC 中的默认子网。
  - 代理创建后，所使用的子网无法更改。
  - 对于集群和 active/standby 代理，子网必须位于不同的可用区中。
  - 对于单实例代理，您可以指定要使用的子网，代理将在同一可用区内创建。
- 安全组
  - 如果未提供安全组，将使用默认 VPC 中的默认安全组。
  - 单实例、集群和 active/standby 代理至少需要一个安全组（例如，默认安全组）。

> **Note**
>
> 公共 RabbitMQ 代理不使用子网或安全组。

- 代理创建后，所使用的安全组无法更改。安全组本身仍可修改。

在中配置私人经纪人 AWS 管理控制台

要配置私有代理，请首先在 AWS 管理控制台中创建新代理。然后，在网络设置 部分，按以下步骤配置代理的连接性：

1. 为您的代理选择私有访问。要连接到私人代理，可以使用 IPv4 IPv6、或双栈（IPv4 和 IPv6）。有关更多信息，请参阅 Connecting to Amazon MQ。
2. 接下来，选择使用默认 VPC、子网和安全组，或选择选择现有 VPC、子网和安全组。如果您不希望使用默认或现有的 VPC、子网或安全组，则必须创建一个新的才能连接到私有代理。

> **Note**
>
> 对于私有代理访问，连接方法将与所选子网的 IP 类型相同。代理创建后，VPC 端点将无法更改，并且始终具有所选子网的 IP 类型。如果您想使用新的 IP 类型，则必须创建新的代理。

> **Note**
>
> Amazon MQ for ActiveMQ 不使用 VPC 端点。当您首次创建 ActiveMQ 代理时，Amazon MQ 会在 VPC 中预置一个弹性网络接口（ENI）。安全组位于 ENI 中，并且可用于公共和私有代理。

访问不可公开访问的 Amazon MQ 代理 Web 控制台

当您关闭经纪商的公开访问权限时，创建经纪人的 AWS 账户 ID 可以访问私人经纪商。如果关闭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。

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. 安装代理客户端，例如FoxyProxy在您的计算机上。
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。

> **Important**
>
> 如果使用的是私有代理，则可能会看到未在 VPC 中配置的 IP 地址。这些 IP 地址来自 Amazon MQ 基础设施上的 RabbitMQ，无需任何操作。

计划 Amazon MQ 代理的维护时段

在维护时段，Amazon MQ 会定期对消息代理的硬件、操作系统或引擎软件进行维护。例如，如果您更改了代理实例类型，Amazon MQ 将在下一个计划维护窗口中应用您的更改。维护的持续时间最长可达两小时，具体取决于为消息代理安排的操作。通过选择跨多个可用区（AZ）且具有高可用性的代理部署模式，可以最大限度地减少维护时段内的停机时间。

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 一次只执行一个实例的维护操作，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

在一周内。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 通过始终保持至少两个运行节点，每次只对一个节点执行维护操作。

首次创建代理时，可以将维护时段安排在每周的指定时间执行一次。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。代理维护时段结束后，Amazon MQ 会重置限制，您可以在下一个维护时段出现前再次调整计划。调整代理维护时段时不会影响代理的可用性。

要调整代理维护时段，您可以使用AWS 管理控制台、AWS CLI 或 Amazon MQ API。

使用AWS 管理控制台计划代理维护时段

使用AWS 管理控制台调整代理维护时段

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. 接下来，选择计划修改。然后选择下次重启后或立即。选择 After the next reboot 将立即更新维护窗口，而无需重启代理。选择立即将立即重启代理。
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。

使用 AWS CLI 计划代理维护时段

使用 AWS CLI 调整代理维护时段

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
```

使用 Amazon MQ API 计划代理维护时段

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 Amazon MQ 端点的更多信息，请参阅 AWS 一般参考 中的 Amazon MQ 端点和配额。

```
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 DescribeBroker API 操作来验证维护时段是否已成功更新。broker-id 已指定为路径参数。

```
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
```

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 管理控制台重启 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. 从经纪人列表中，选择您的经纪商名称（例如 MyBroker）。
3. 在***MyBroker***页面上，选择操作，重启代理。

> **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 管理控制台删除代理。

删除 Amazon MQ 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。

Amazon 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	说明
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。
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。通过从问题排查中的列表内选择操

控制台	API	说明
		作所需代码，您可以找到解决问题的说明。

为 Amazon MQ 资源添加标签

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	键	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 控制台中添加标签

通过以下步骤，您可以在 Amazon MQ 控制台中快速为您正在创建的资源添加标签。

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)。

添加标签后，您可以在 Amazon MQ 控制台中查看、编辑和删除资源的标签。您还可以使用 REST API 查看资源的标签。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 管理控制台、Amazon MQ REST API 或 AWS Command Line Interface 创建、管理和删除代理。

Amazon MQ for ActiveMQ 代理可作为单实例代理或主动/备用代理进行部署。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
- 基于 WebSocket 的 MQTT
- OpenWire
- STOMP
- 基于 WebSocket 的 STOMP

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

Amazon MQ for ActiveMQ 代理

什么是 Amazon MQ for 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`) 和大小 (`large`, `medium`) 的组合描述称为代理实例类型（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 Amazon MQ for ActiveMQ 代理的部署选项。

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：

- 入门：创建并连接 ActiveMQ 代理
- 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
- 基于 WebSocket 的 MQTT
- OpenWire
- STOMP
- 基于 WebSocket 的 STOMP

属性

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 终端节点。

  - `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。

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

- Authorization*
- 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 ActiveMQ 代理用户
- Users

用户属性

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

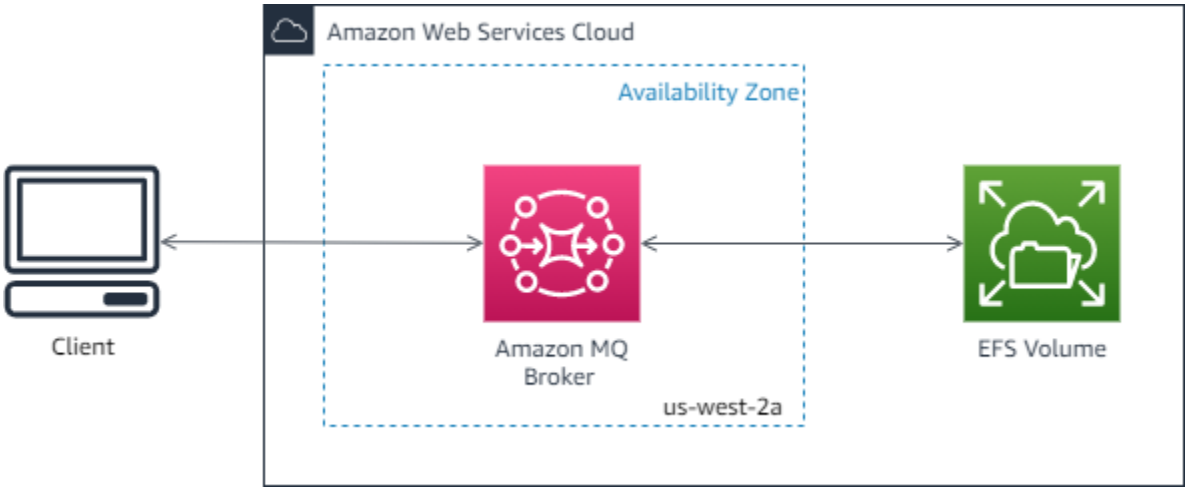
Amazon MQ for ActiveMQ 代理的部署选项

Amazon MQ 为代理提供单实例和集群部署选项。

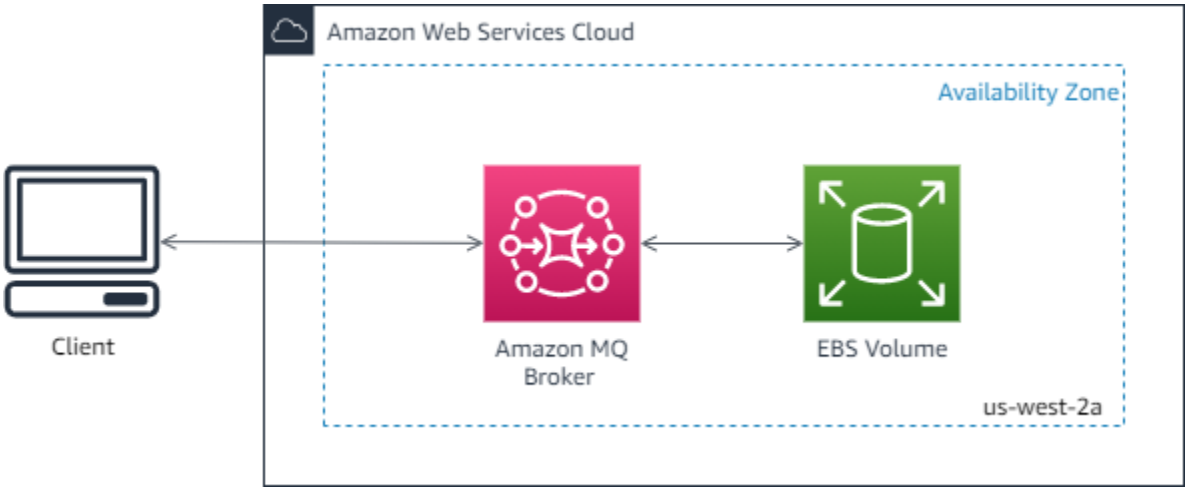
选项 1：Amazon MQ 单实例代理

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冗余存储数据来提供最高级别的耐久性和可用性 (AZs)。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。

下图说明了一个单实例代理，其中包含跨多个 AZs实例复制的 Amazon EFS 存储。



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。



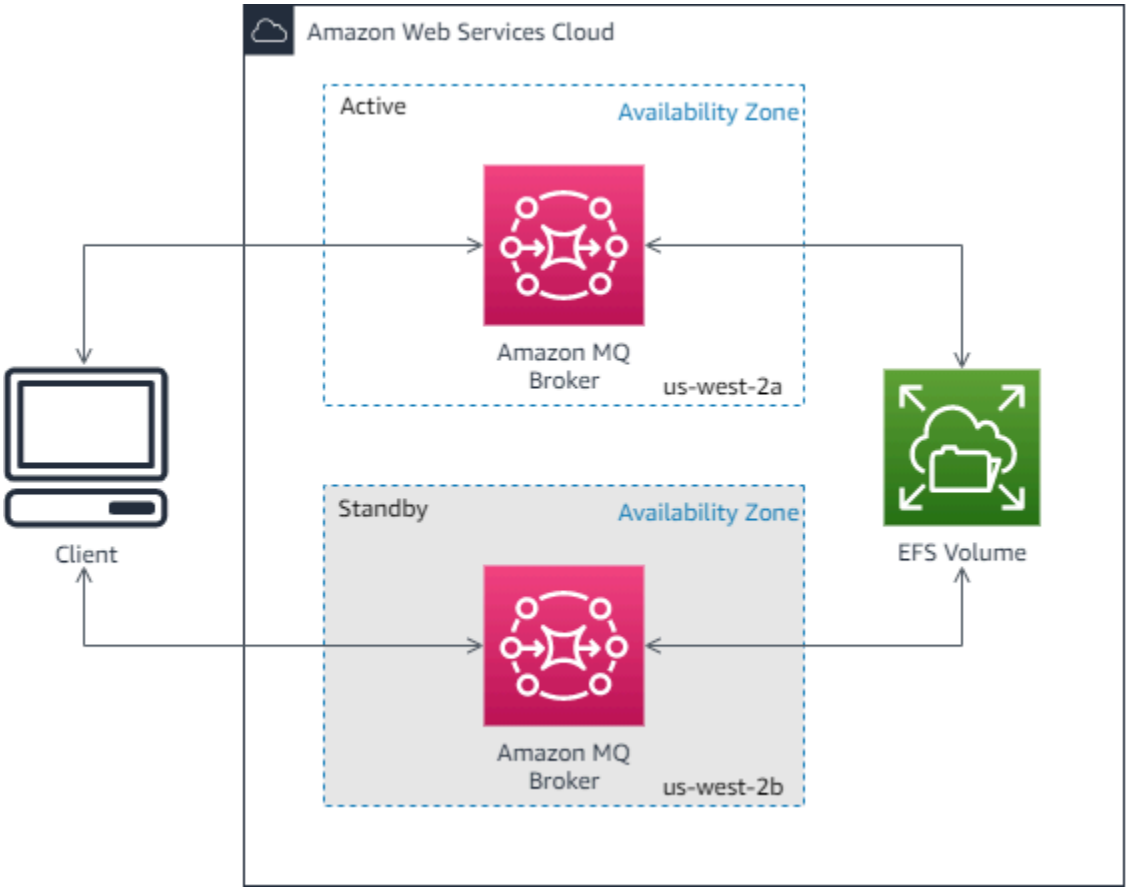
选项 2：亚马逊 MQ active/standby 代理以实现高可用性

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冗余存储数据来提供最高级别的耐久性和可用性 (AZs)。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。您启动的维护窗口和代理重启将导致失效转移发生。当您重启代理时，故障转移仅需几秒钟。

对于 active/standby 代理商，Amazon MQ 提供了两个 ActiveMQ Web 控制台 URLs，但一次只有一个 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。对于线路级协议端点，您应允许您的应用程序使用失效转移传输连接到任一端点。

下图说明了在多个 active/standby 代理之间复制 Amazon EFS 存储 AZs。



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跃的单实例代理或 active/standby 代理组成。创建代理网络可以通过多个代理实例提高可用性、容错能力和负载平衡。

代理网络如何工作？

代理网络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。网络连接器提供从一个代理到另一个代理的按需消息传递。网络连接器在代理配置中配置为非双工或双工连接。对于非双工连接，消息仅从一个代理转发到另一个代理。对于双工连接，消息在两个代理之间双向转发。

如果网络连接器配置为双工，消息也会从 Broker2 转发到 Broker1。

您可以在代理网络中同时使用非双工和双工连接。您可能希望引入到另一个代理的双工连接以改善流量，或避免限制增加。双工连接对于从本地到 Amazon MQ 托管代理的部分迁移也很有用。

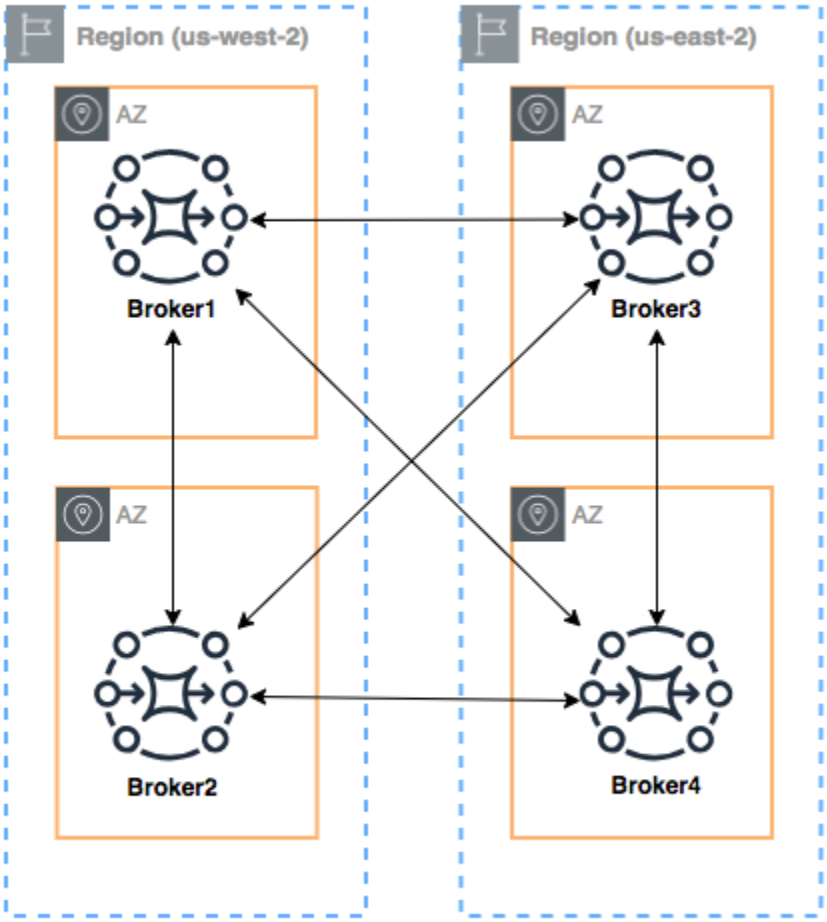
代理网络如何处理凭据？

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。

跨区域

要配置跨 AWS 区域的代理网络，请在这些区域部署代理，然后为这些代理的端点配置网络连接器。



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

```
</networkConnectors>
```

Broker2 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-
west-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)
```

当新的经纪商连接时，它将收到网络 URIs 中所有经纪人的列表。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 代理实例类型

代理实例类 (m5) 和大小 (large, medium) 的组合描述称为代理实例类型（例如，mq.m5.large）。下表列出了可用于 ActiveMQ 代理的 Amazon MQ 代理实例类型。

Amazon MQ 会在实例类型支持终止日期前至少提前 90 天发出通知。我们建议在支持终止日期之前将代理升级到新的实例类型，以防止任何中断。

> **Important**
>
> 2025 年 3 月 17 日之后，您无法在 t2.micro 或 mq.m4.large 上创建代理。

实例类型	vCPU	内存 (GiB)	推荐使用	存储	Amazon MQ 终止支持
mq.t3.micro	2	1	评估	EFS	
mq.m5.large	2	8	生产	EFS 或 EBS	
mq.m5.xlarge	4	16	生产	EFS 或 EBS	
mq.m5.2xlarge	8	32	生产	EFS 或 EBS	

实例类型	vCPU	内存 (GiB)	推荐使用	存储	Amazon MQ 终止支持
mq.m5.4xlarge	16	64	生产	EFS 或 EBS	

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 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。

> **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。
>
> 您只能使用 DeleteConfiguration API 删除配置。更多信息，请参阅 Amazon MQ API 参考中的配置。

属性

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

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

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 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。您可以立即应用或在维护时段内应用配置。

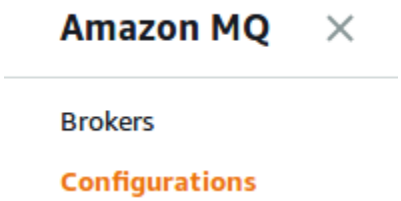
以下示例演示如何使用 AWS 管理控制台创建和应用 Amazon MQ 代理配置。

> **Important**
>
> 您只能使用 `DeleteConfiguration` API 删除配置。更多信息，请参阅 Amazon MQ API 参考中的配置。

创建新的配置

要创建新的代理配置，首先要创建新配置。

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. 在创建配置页面的详细信息部分，输入配置名称（例如 `MyConfiguration`）并选择代理引擎版本。

> **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

创建代理配置后，需要使用配置修订版编辑配置。

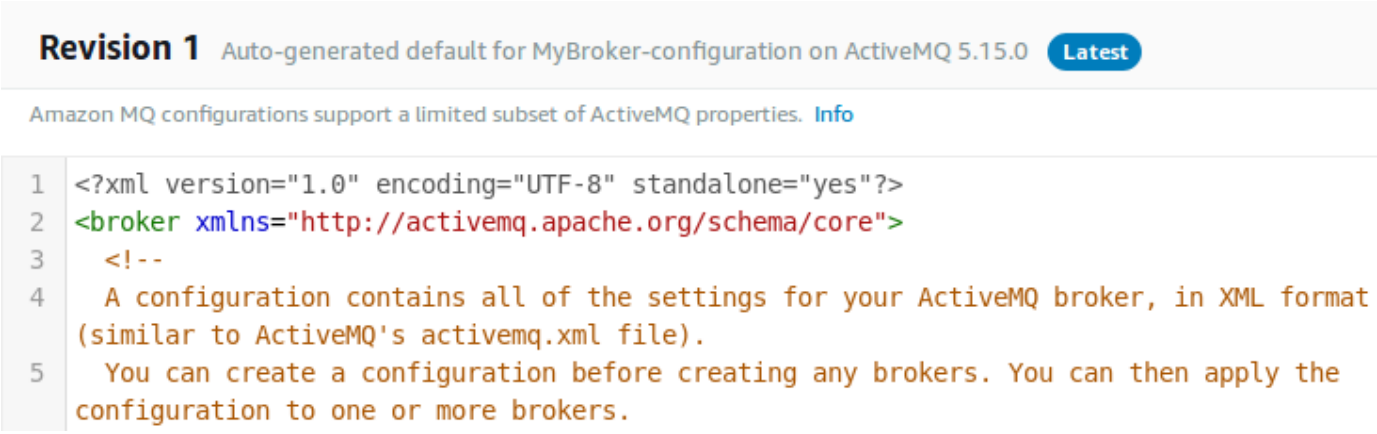
1. 从配置列表中选择***`MyConfiguration`***。

> **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。

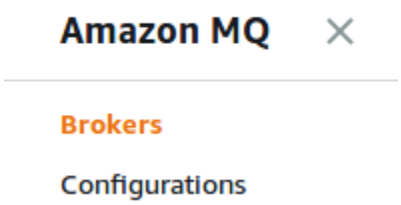
   将会保存配置的新修订。

> **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。

将配置修订应用到代理

修订配置后，可以将配置修订版应用到代理。

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**
>
> 单实例代理在重启期间处于离线状态。对于集群代理，在代理重启期间，每次只有一个节点宕机。

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 Amazon MQ for ActiveMQ 配置修订版

将配置修订版应用到代理后，您可能需要对其进行编辑。按照以下说明编辑配置修订版。

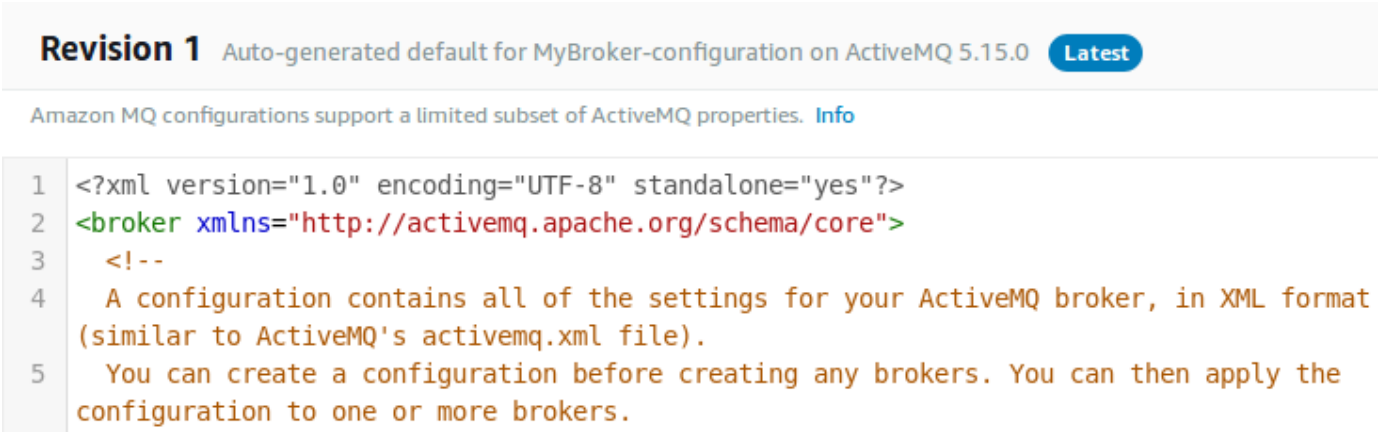
1. 登录 Amazon MQ 控制台。
2. 从经纪人列表中，选择您的经纪商（例如 MyBroker），然后选择编辑。
3. 在***MyBroker***页面上，选择 “编辑”。
4. 在 “编辑 ***MyBroker***” 页面的 “配置” 部分，选择一个配置和一个修订版，然后选择编辑。

> **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。

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。

Element
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)
constantPendingMessageLimitStrategy (属性)
discarding (属性)
discardingDLQBrokerPlugin (属性)
fileCursor
fileDurableSubscriberCursor

Element
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)
priorityDispatchPolicy
priorityNetworkDispatchPolicy
queryBasedSubscriptionRecoveryPolicy (属性)
queue (属性)
redeliveryPlugin (属性 | 子集合元素)

Element
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)
timeStampingBrokerPlugin (属性)
topic (属性)
transportConnector (属性)
uniquePropertyMessageEvictionStrategy (属性)
virtualDestinationInterceptor (子集合元素)

Element
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。

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

Element	属性
	maxSlowDuration
	name
authorizationEntry	admin
	queue
	read
	tempQueue
	tempTopic
	topic
	write
broker	advisorySupport
	allowTempAutoCreationOnSend
	cacheTempDestinations
	consumerSystemUsagePortion
	dedicatedTaskRunner
	deleteAllMessagesOnStartup
	keepDurableSubsActive
	enableMessageExpirationOnActiveDurableSubs
	maxPurgedDestinationsPerSweep
	maxSchedulerRepeatAllowed
	monitorConnectionSplits

Element	属性
	networkConnectorStartAsync
	offlineDurableSubscriberTaskSchedule
	offlineDurableSubscriberTimeout
	persistenceThreadPriority
	persistent
	populateJMSXUserID
	producerSystemUsagePortion
	rejectDurableConsumers
	rollbackOnlyOnAsyncException
	schedulePeriodForDestinationPurge
	schedulerSupport
	splitSystemUsageForProducersConsumers
	taskRunnerPriority
	timeBeforePurgeTempDestinations
	useAuthenticatedPrincipalForJMSXUserID
	useMirroredQueues
	useTempMirroredQueues
	useVirtualDestSubs

Element	属性
	useVirtualDestSubsOnCreation
	useVirtualTopics
cachedMessageGroupMapFactory	cacheSize
compositeQueue	concurrentSend
	copyMessage
	forwardOnly
	name
	sendWhenNotMatched
compositeTopic	concurrentSend
	copyMessage
	forwardOnly
	name
	sendWhenNotMatched
conditionalNetworkBridgeFilterFactory	rateDuration
	rateLimit
	replayDelay
	replayWhenNoConsumers
	selectorAware 在以下版本中受支持： Apache ActiveMQ 5.16.x

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

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

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

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

Element	属性
	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
	useCache
	useConsumerPriority
usePrefetchExtension	
prefetchRatePendingMessageLimitStrategy	multiplier

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

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

Element	属性
	processNetworkMessages
	ttlCeiling
topic	DLQ
	physicalName
transportConnector	name
	updateClusterClients
	rebalanceClusterClients
	updateClusterClientsOnRemove
uniquePropertyMessageEvictionStrategy	evictExpiredMessagesHighWatermark
	propertyName
virtualTopic	concurrentSend
	local
	dropOnResourceLimit
	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。

属性

networkConnectionStart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。

Element	子集合元素	子元素
authorizationMap	authorizationEntries	authorizationEntry
		tempDestinationAuthorizationEntry
	defaultEntry	authorizationEntry
		tempDestinationAuthorizationEntry

Element	子集合元素	子元素
	tempDestinationAuthorizationEntry	tempDestinationAuthorizationEntry
authorizationPlugin	map	authorizationMap
broker	destinationInterceptors	mirroredQueue
		virtualDestinationInterceptor
	destinationPolicy	policyMap
	destinations	queue
		tempQueue
		tempTopic
		topic
	networkConnectors	networkConnector
	persistenceAdapter	kahaDB
	plugins	authorizationPlugin
		discardingDLQBrokerPlugin
		forcePersistencyModeBrokerPlugin
		redeliveryPlugin
		statisticsBrokerPlugin
		timeStampingBrokerPlugin

Element	子集合元素	子元素
	systemUsage	systemUsage
	transportConnector	name
		updateClusterClients
		rebalanceClusterClients
		updateClusterClientsOnRemove
compositeQueue	forwardTo	queue
		tempQueue
		tempTopic
		topic
		filteredDestination
compositeTopic	forwardTo	queue
		tempQueue
		tempTopic
		topic
		filteredDestination
policyEntry	deadLetterStrategy	discarding
		individualDeadLetterStrategy
		sharedDeadLetterStrategy

Element	子集合元素	子元素
	destination	queue
		tempQueue
		tempTopic
		topic
	dispatchPolicy	priorityDispatchPolicy
		priorityNetworkDispatchPolicy
		roundRobinDispatchPolicy
		simpleDispatchPolicy
		strictOrderDispatchPolicy
		clientIdFilterDispatchPolicy
	messageEvictionStrategy	oldestMessageEvictionStrategy
		oldestMessageWithLowestPriorityEvictionStrategy
		uniquePropertyMessageEvictionStrategy
	messageGroupMapFactory	cachedMessageGroupMapFactory

Element	子集合元素	子元素
		messageGroupHashBucketFactory
		simpleMessageGroupMapFactory
	pendingDurableSubscriberPolicy	fileDurableSubscriberCursor
		storeDurableSubscriberCursor
		vmDurableCursor
	pendingMessageLimitStrategy	constantPendingMessageLimitStrategy
		prefetchRatePendingMessageLimitStrategy
	pendingQueuePolicy	fileQueueCursor
		storeCursor
		vmQueueCursor
	pendingSubscriberPolicy	fileCursor
		vmCursor
	slowConsumerStrategy	abortSlowAckConsumerStrategy
		abortSlowConsumerStrategy

Element	子集合元素	子元素
	subscriptionRecoveryPolicy	fixedCountSubscriptionRecoveryPolicy
		fixedSizedSubscriptionRecoveryPolicy
		lastImageSubscriptionRecoveryPolicy
		noSubscriptionRecoveryPolicy
		queryBasedSubscriptionRecoveryPolicy
		retainedMessageSubscriptionRecoveryPolicy
timedSubscriptionRecoveryPolicy		
policyMap	defaultEntry	policyEntry
	policyEntries	policyEntry
redeliveryPlugin	redeliveryPolicyMap	redeliveryPolicyMap
redeliveryPolicyMap	defaultEntry	redeliveryPolicy
	redeliveryPolicyEntries	redeliveryPolicy
retainedMessageSubscriptionRecoveryPolicy	wrapped	fixedCountSubscriptionRecoveryPolicy
		fixedSizedSubscriptionRecoveryPolicy

Element	子集合元素	子元素
		lastImageSubscriptionRecoveryPolicy
		noSubscriptionRecoveryPolicy
		queryBasedSubscriptionRecoveryPolicy
		retainedMessageSubscriptionRecoveryPolicy
		timedSubscriptionRecoveryPolicy
sharedDeadLetterStrategy	deadLetterQueue	queue
		tempQueue
		tempTopic
		topic
virtualDestinationInterceptor	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>
```

> **Note**
>
> ActiveMQ on Amazon MQ 中的 `activemq-webconsole` 组拥有所有队列和主题的管理权限。该组中的所有用户都有管理员访问权限。

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，代理常用的用户名是myCommonUser，连接是duplex， OpenWire 端点 URI 的前缀为static:，表示代理之间存在 one-to-one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，代理常用的用户名是，连接是 myCommonUserduplex，逗号分隔的 OpenWire端点列表以逗号分隔 URIs 为前缀masterslave:，表示代理之间存在故障转移连接。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-west-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`

journalDiskSync间隔

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`

journalDiskSync策略

> 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>
```

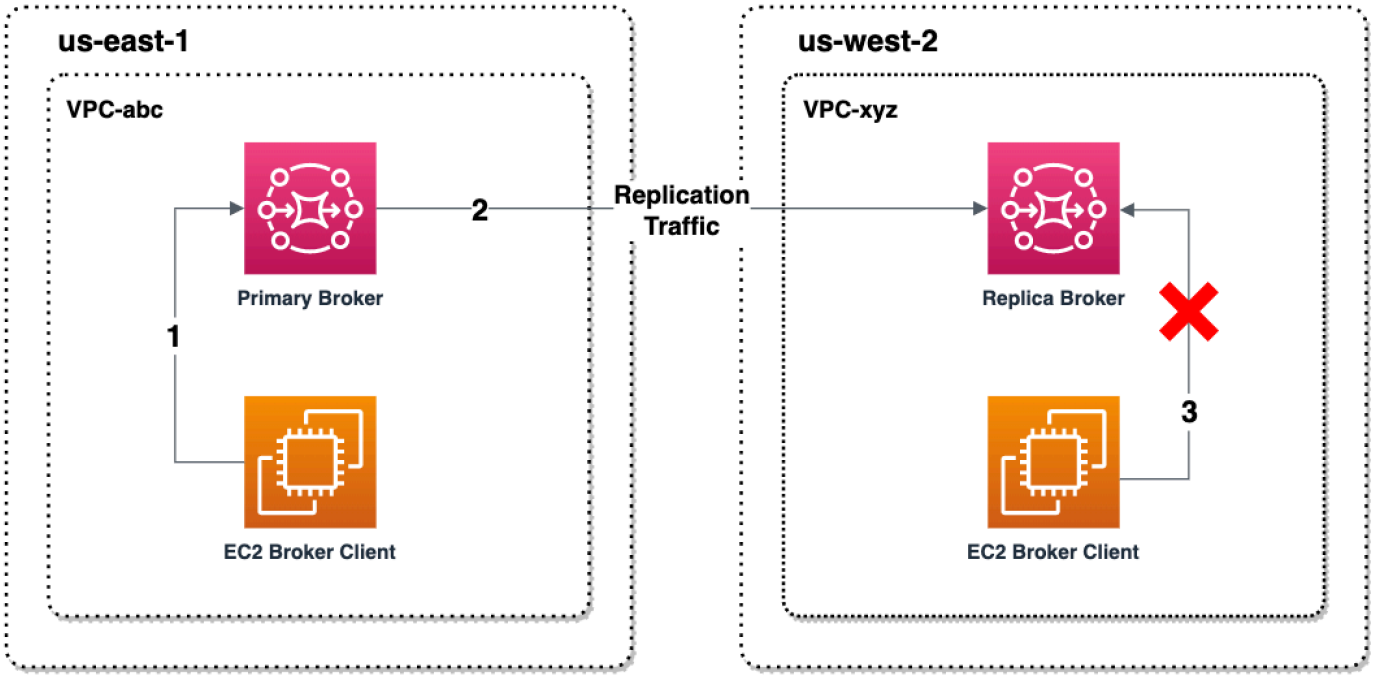
Amazon MQ for ActiveMQ 的跨区域数据复制

Amazon MQ for ActiveMQ 提供跨区域数据复制 (CRDR) 功能，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 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。

用于跨区域数据复制的主代理和副本代理

您可以创建主代理和副本代理，以便将数据从主区域的主代理异步复制到副本 AWS 区域的副本代理。主区域由一对冗余的经纪商组成，这些 active/standby 经纪人被称为主经纪商。辅助区域由一对冗余的 active/standby 代理组成，称为副本代理。

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 Amazon MQ 跨区域数据复制代理。

> **Note**
>
> 仅适用于 active/standby 经纪人。
> 不适用于镜像队列。

创建 Amazon MQ 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，请参阅用于跨区域数据复制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 Amazon MQ for ActiveMQ 代理的部署选项。
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，请参阅入门：创建并连接 ActiveMQ 代理

删除 Amazon MQ 跨区域数据复制代理

要删除主跨区域数据复制（CRDR）代理或副本，必须先取消配对，然后重启代理。以下说明说明如何使用 AWS 管理控制台取消配对并重新启动代理。

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“confirm”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。

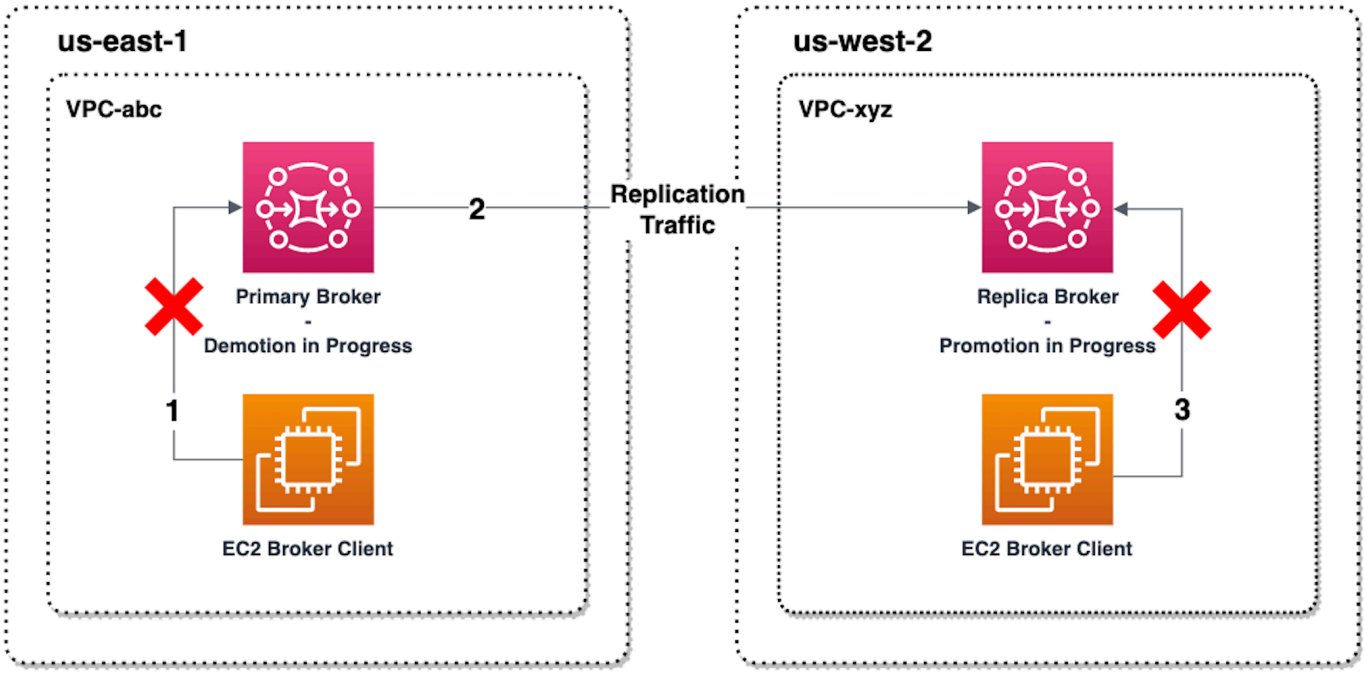
启动切换或失效转移以将 Amazon MQ 副本代理提升为主代理角色

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。

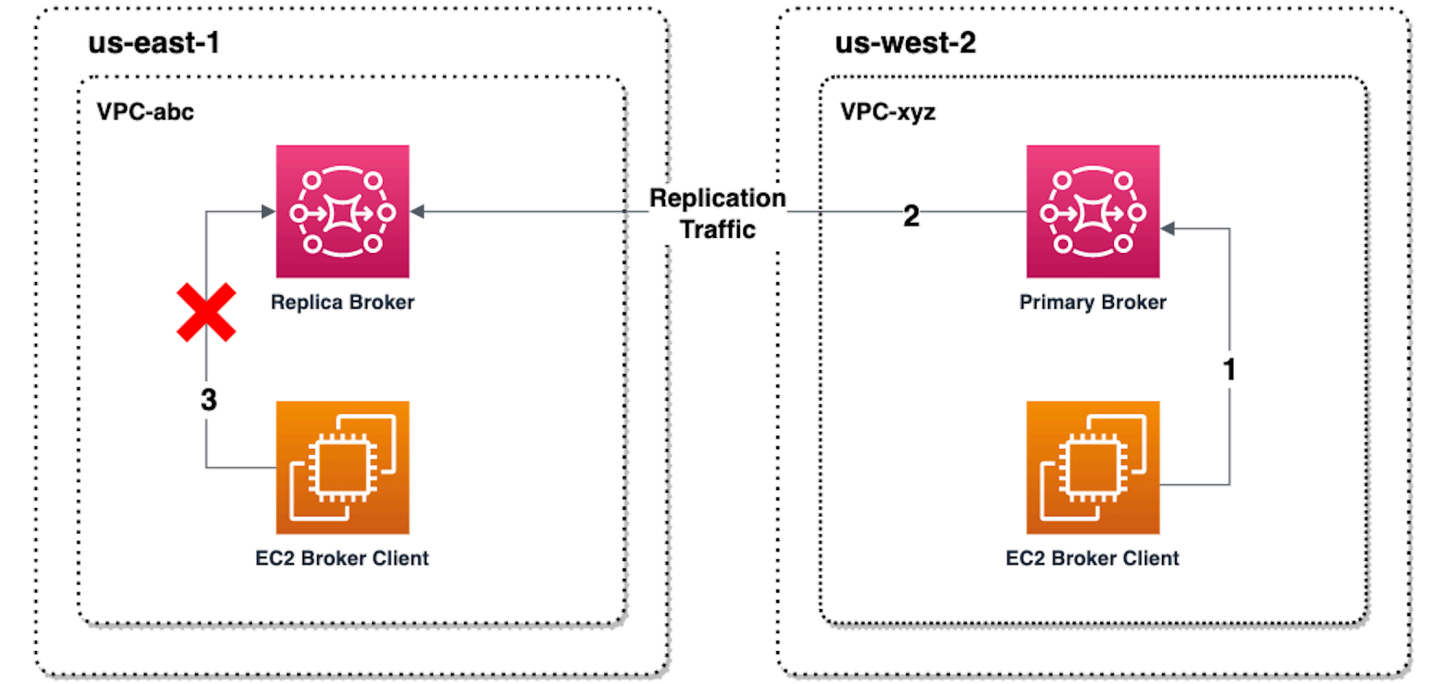
切换优先考虑一致性而不是可用性。当此失效转移操作完成时，可以保证代理处于相同的状态。通过切换，在建立代理间一致性的同时，可能会出现两个代理都无法进行客户端连接的时期。在提升副本的瞬间，这两个代理将处于相同的状态。切换是否成功取决于两个区域的运行状况和区域间网络。

失效转移优先考虑可用性而不是一致性。当此操作完成时，不能保证代理处于相同的状态。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，主代理的 VPC 中的客户端在操作正在进行期间无法发生进一步的状态变化，并且主代理降级为副本。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，请参阅访问亚马逊 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。	

Amazon MQ CloudWatch 指标	使用 CRDR 的原因
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。

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 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
- 步骤 3：（可选）Connect 到 AWS Lambda 函数
- 创建 ActiveMQ 代理用户
- 编辑 ActiveMQ 代理用户
- 删除 ActiveMQ 代理用户
- 将 Java Message Service（JMS）与 ActiveMQ 配合使用的有效示例

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。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`
- `networkConnectionStart##`
- 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，请参阅入门：创建并连接 ActiveMQ 代理。
- 这两个代理必须位于同一 VPC 中或处于对等状态。 VPCs有关更多信息 VPCs，请参阅什么是 Amazon VPC？ 在 Amazon VPC 用户指南和什么是 VPC 对等互连？ 在《亚马逊 VPC 对等互连指南》中。

> **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 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-
west-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)。

   在网络桥接部分，`MyBroker2` 的名称和地址列在远程代理和远程地址列中。
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

接下来，为您的应用程序启用入站连接。

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

使用以下说明创建消息生产者并接收消息。

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

> **Note**
>
> 对于 active/standby 代理商，Amazon MQ 提供了两个 ActiveMQ Web 控制台 URLs，但一次只有一个 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 Amazon MQ for ActiveMQ 代理的部署选项）。
> 对于线路级协议端点，您应允许您的应用程序使用失效转移传输连接到任一端点。

```
// 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

使用以下说明创建消息生产者并接收消息。

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**
>
> Amazon MQ 不支持由私人 CA 签发的服务器证书。

您可以使用以下启用 TLS 的协议来访问 ActiveMQ 代理：

- AMQP
- MQTT
- MQTT 结束了 WebSocket
- OpenWire
- STOMP
- 大吃一惊 WebSocket

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Users。

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。

> **Important**
>
> 使用 LDAP 时，认证不区分大小写，但授权对您的用户名区分大小写。

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)。

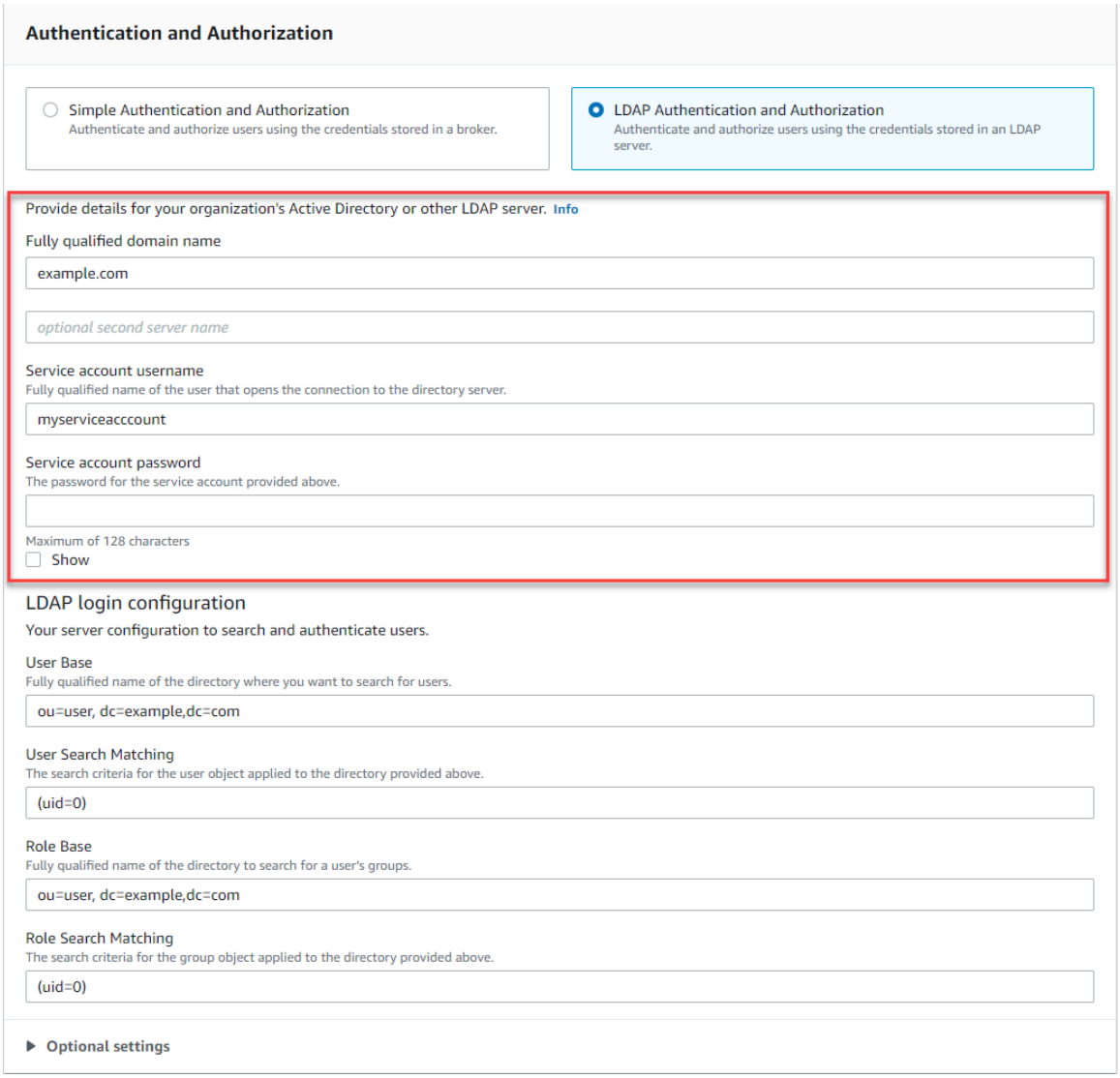
提供有关服务账户的以下信息：

- 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。

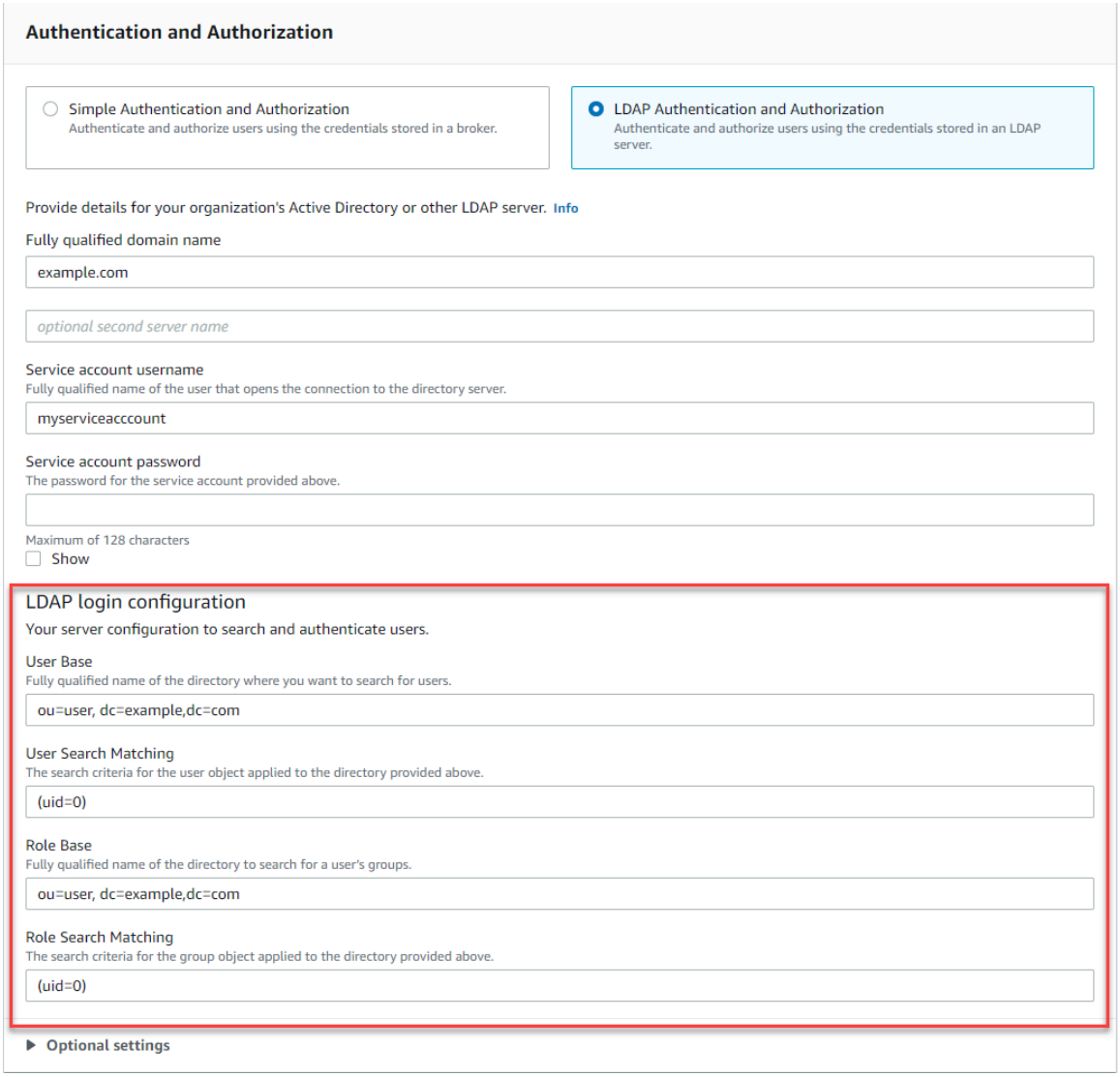


在 LDAP login configuration (LDAP 登录配置) 部分中，提供以下必要信息：

- 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：

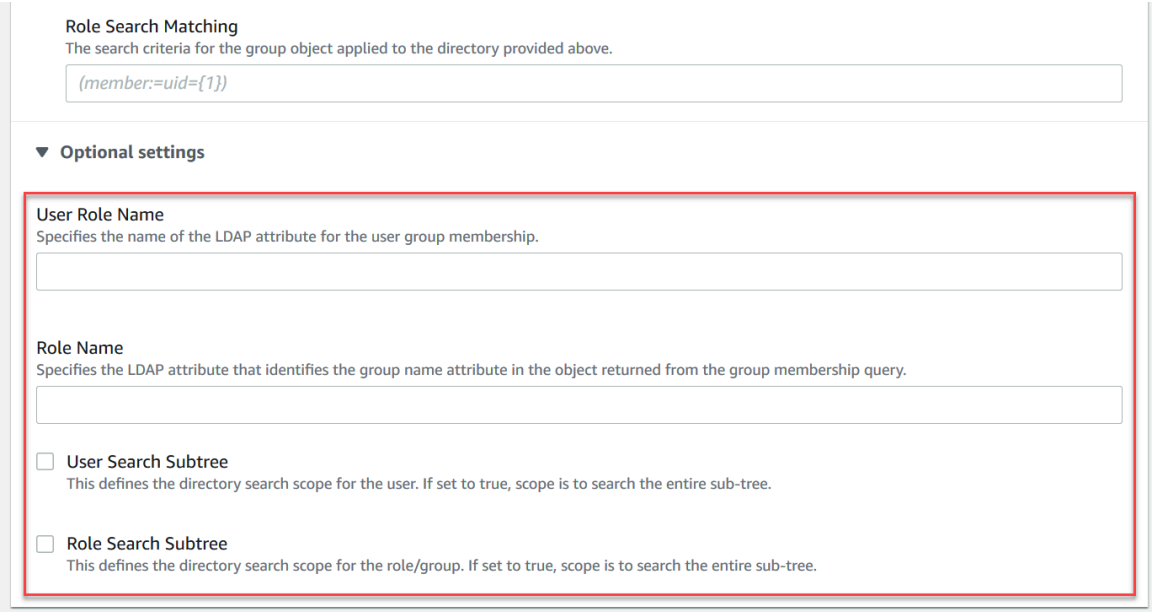
```
dn: uid=jdoe,ou=user,dc=example,dc=com
objectClass: user
uid: jdoe
```

```
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。

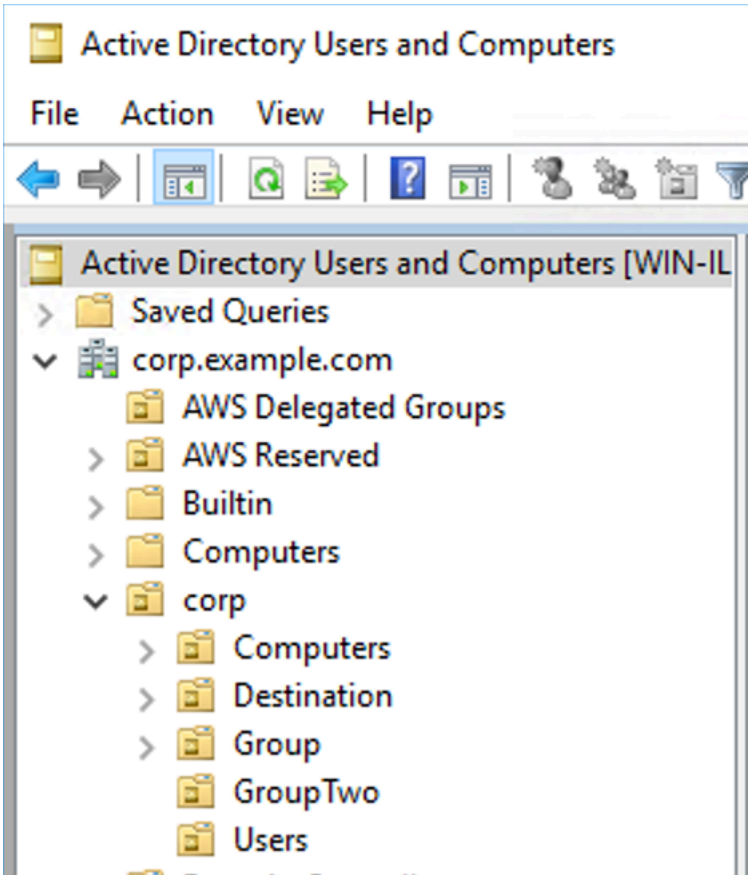
身份验证

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，如果您正在使用 AWS Managed Microsoft AD，并且有域组件`corp`和`example`、`com`，并且在这些组件中包含组织单位`corp`和`Users`，则应使用以下内容作为用户群：

```
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**
>
> 为了能够使用代理`activemq.xml`配置文件中的`cachedLDAPAuthorizationMap`元素，您必须在通过创建配置时选择 LDAP 身份验证和授权选项 AWS 管理控制台，或者通过设置创建配置 AWS 管理控制台，或者在使用 Amazon MQ API 创建新配置LDAP时将`authenticationStrategy`属性设置为。

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`
>
> 当您创建经纪商时，Amazon MQ 会使用您通过或您的 API 请求的`ldapServerMetadata`属性中提供的值来代替上述属性。 AWS 管理控制台

以下演示了 `cachedLdapAuthorizationMap` 的工作示例。

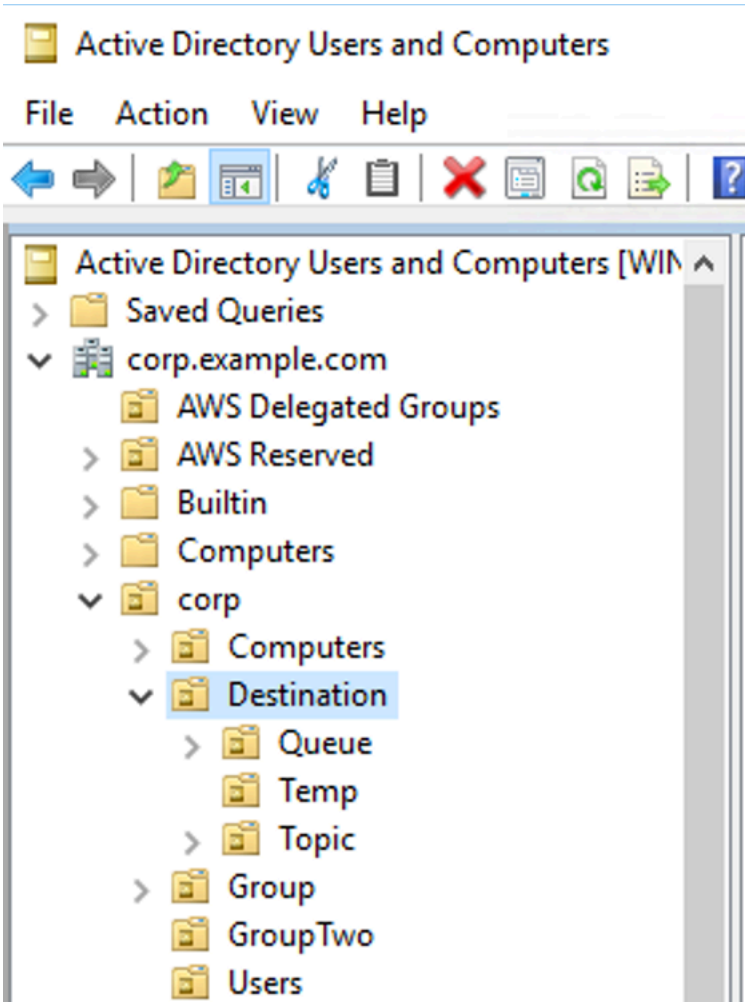
```
<authorizationPlugin>
    <map>
```

```
        <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，对于上面的示例 AWS Managed Microsoft AD，使用corpexample、和的相同域组件com，您可以指定一个名为的组织单位destination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：

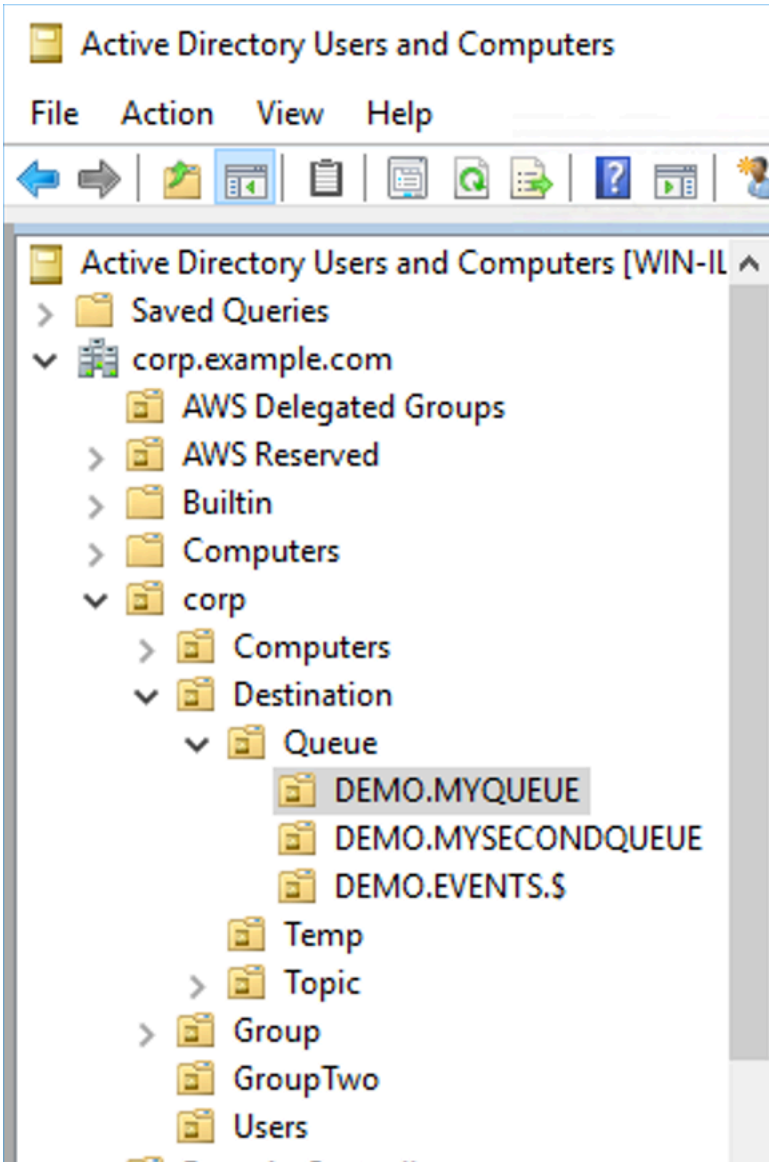
```
OU=DEMO.EVENTS.$,OU=Queue,OU=Destination,OU=corp,DC=corp,DC=example,DC=com
```

> **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：

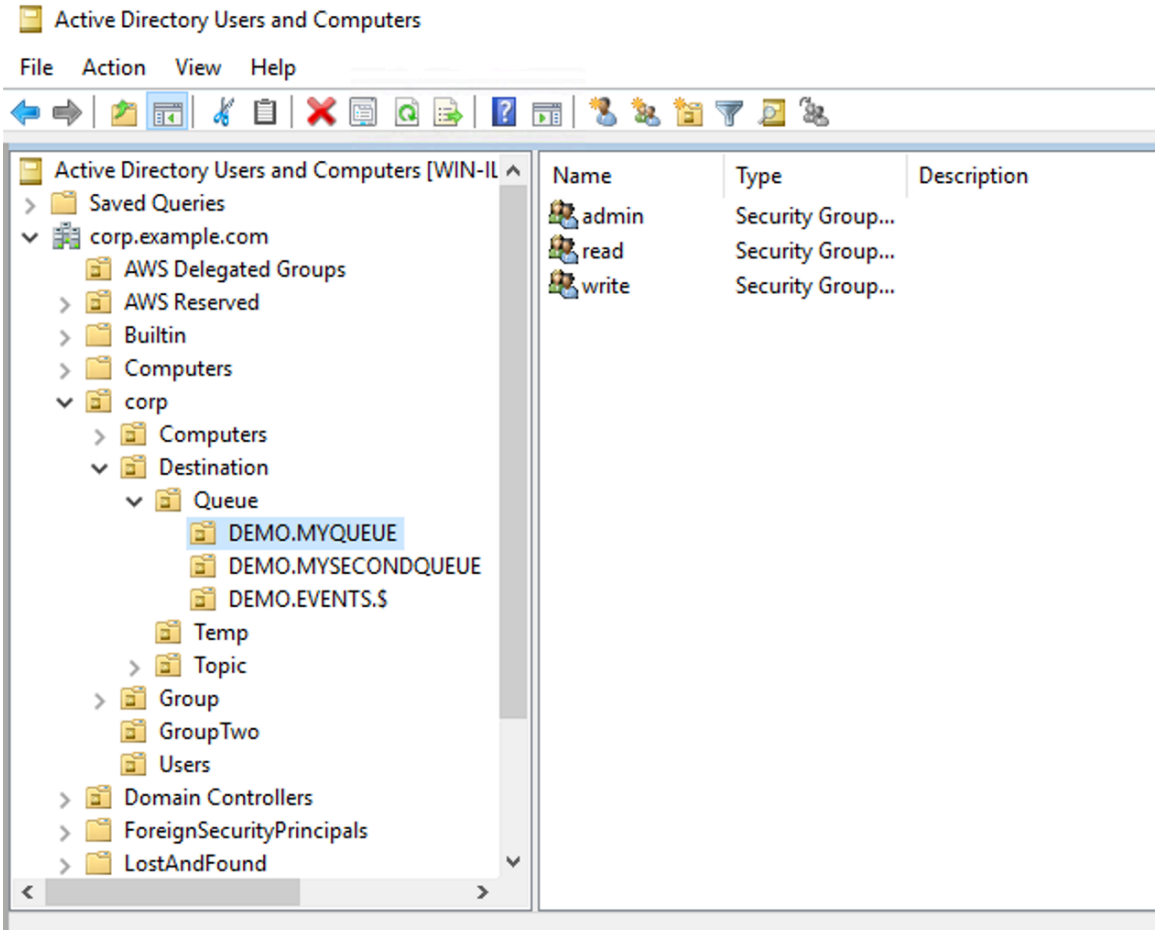
```
OU=DEMO.MYQUEUE,OU=Queue,OU=Destination,OU=corp,DC=corp,DC=example,DC=com
```



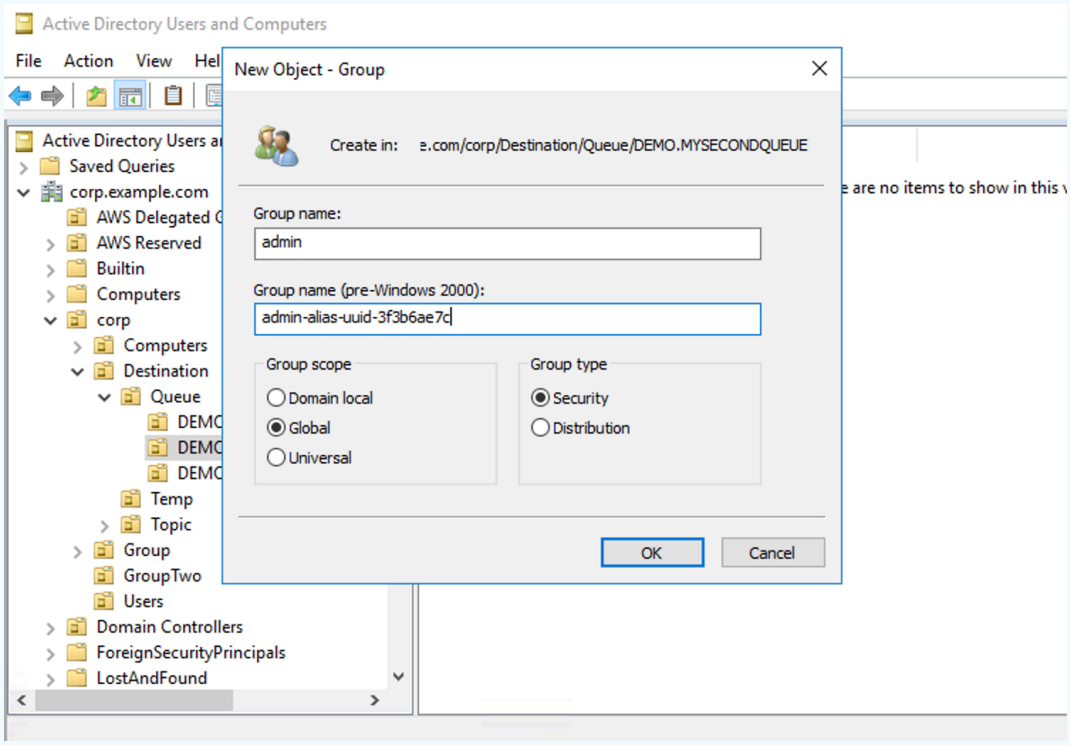
安全组

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，这些都是权限。read/write/admin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。

步骤 3：（可选）Connect 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 管理控制台为 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。

创建 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。

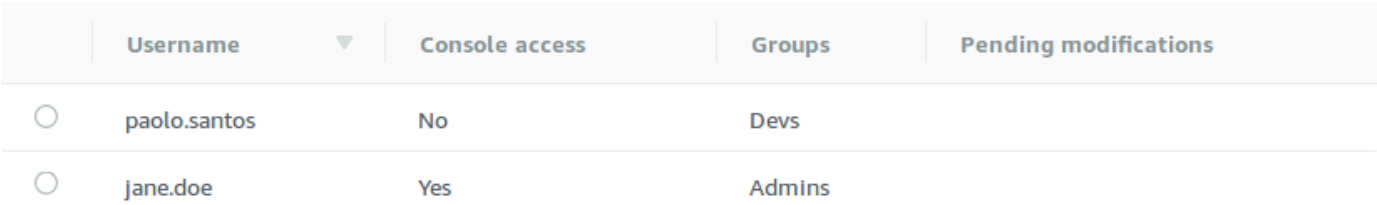
> **Note**
>
> ActiveMQ on Amazon MQ 中的 `activemq-webconsole` 组拥有所有队列和主题的管理权限。该组中的所有用户都有管理员访问权限。

以下示例演示如何使用 AWS 管理控制台创建、编辑和删除 Amazon MQ 代理用户。

创建新的 ActiveMQ 代理用户

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。

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。

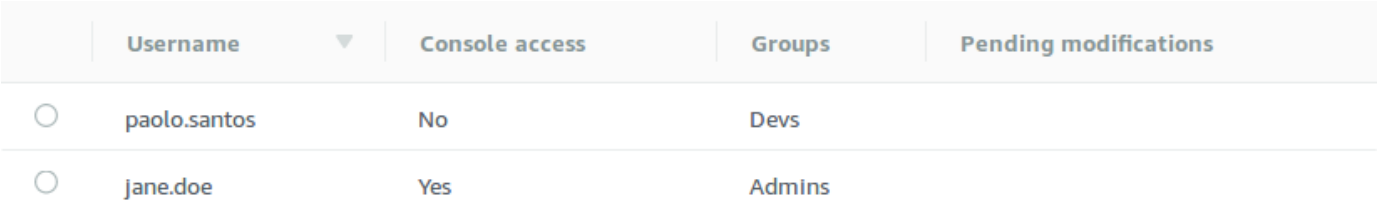
> **Important**
>
> 对用户进行更改不会立即将更改应用于用户。要应用更改，必须等待下一维护时段或者重启代理。

编辑 ActiveMQ 代理用户

要编辑现有用户，请执行以下操作：

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)。

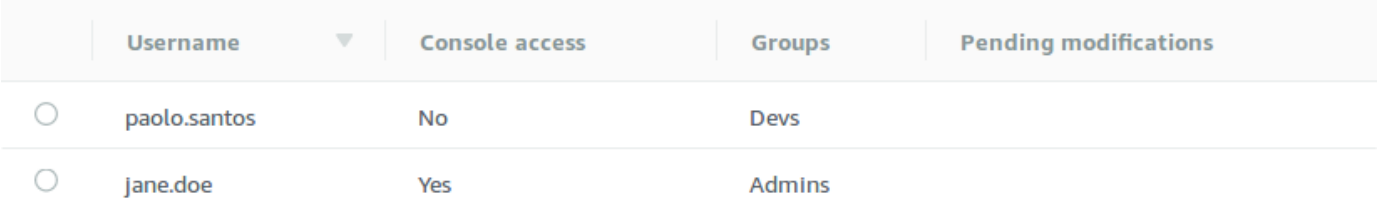
> **Important**
>
> 对用户进行更改不会立即将更改应用于用户。要应用更改，必须等待下一维护时段或者重启代理。

删除 ActiveMQ 代理用户

不再需要某个用户时，可以删除该用户。

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。

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

要以编程方式使用 Amazon MQ，必须使用入站连接。

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。

Amazon MQExample .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);
                        }
                    }
```

```
}
```

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 Amazon MQ for ActiveMQ 实施中，`X` 表示主要版本，`Y` 表示次要版本，`Z` 表示补丁版本号。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 5.17 版升级到 6.0 版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变化，则该版本变更被视为次要版本变更。例如，从版本 5 升级。 18 比 5。19 被视为次要版本升级。当 `autoMinorVersionUpgrade` 开启时，Amazon MQ 将您的代理升级到最新的可用补丁版本。

Amazon MQ for ActiveMQ 建议所有代理使用支持的最新次要版本。有关如何升级代理引擎版本的说明，请参阅升级 Amazon MQ 代理引擎版本。

Amazon MQ for ActiveMQ 支持的引擎版本

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。此次升级将在您的经纪商的预定维护时段内，即 end-of-support日期之后的45天内进行。

Amazon MQ 至少会在某个版本终止支持前 90 天发出通知。我们建议您在该 end-of-support日期之前升级您的经纪商，以免出现任何中断。此外，在支持终止日期后 30 天内，您不能在计划终止支持的版本上创建新的代理。

Apache ActiveMQ 版本	Amazon MQ 终止支持
ActiveMQ 5.19（推荐）	
ActiveMQ 5.18	
ActiveMQ 5.17	2025 年 6 月 16 日
ActiveMQ 5.16	2024 年 11 月 15 日
ActiveMQ 5.15	2024 年 9 月 16 日

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。如果在创建代理时没有指定引擎版本号，Amazon MQ 会自动默认为最新的引擎版本号。

引擎版本升级

您可以随时手动将代理升级到下一个受支持的主要版本或次要版本。启用自动次要版本升级后，Amazon MQ 将在维护时段内将代理升级到支持的最新补丁版本。

有关手动升级代理的更多信息，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 `mq.m5.large` 和实例类型的结果，请使用和替换为。*`engine-type`* `ACTIVEMQ` *`instance-type`* `mq.m5.large`

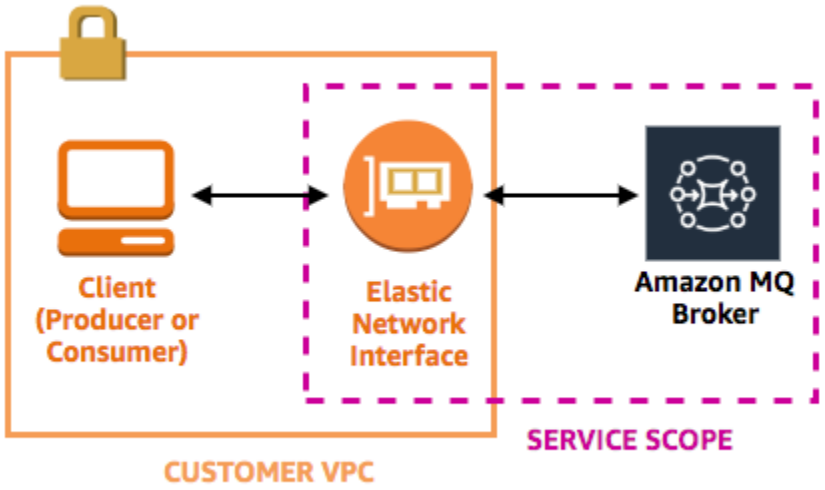
Amazon MQ for ActiveMQ 最佳实践

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。

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 入门：创建并连接 ActiveMQ 代理 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-
west-2.amazonaws.com:61617)?randomize=true
```

> **Important**
>
> 多可用区代理在维护窗口和代理重启期间可能发生失效转移。使用失效转移传输以确保代理可用性。

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 对等互连，请避开 CIDR IPs 范围内的客户端 **10.0.0.0/16**

如果您要在本地基础设施和您的 Amazon MQ 代理之间设置 Amazon VPC 对等连接，则不得在 CIDR 范围内配置客户端连接 IPs 。`10.0.0.0/16`

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。

要监控未处理的准备的 XA 交易，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。

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、Amazon MQ REST API 或创建 AWS 管理控制台、管理和删除经纪人。 AWS Command Line Interface

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。

要了解亚马逊 MQ REST APIs，请参阅亚马逊 M Q REST API 参考。

什么是 Amazon MQ for 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 (`m7g`) 和大小 (`large`, `medium`) 的组合描述称为代理实例类型（ 例如，`mq.m7g.large` ）。

- 单实例代理由 Network Load Balancer (NLB) 后面一个可用区域中的一个代理组成。代理可与您的应用程序和 Amazon EBS 存储卷进行通信。
- 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，请参阅部署 RabbitM Q 代理。

侦听器端口

Amazon MQ 托管 RabbitMQ 代理支持以下侦听器端口，用于应用程序级连接。`amqps`您也可以使用 RabbitMQ Web 控制台和管理 API 将这些端口用于客户端连接。为了安全起见，所有连接都使用 TLS 加密。

- 侦听器端口 5671-用于通过安全 AMQP 网址建立的安全 AMQP 连接。此端口支持 RabbitMQ 4 中的 AMQP 0-9-1 和 AMQP 1.0 协议。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 `amqps` URL：`b-c8352341-ec91-4a78-ad9c-a43f23d325bb.mq.us-west-2.amazonaws.com:5671`。
- 监听器端口443和 15671-您可以互换使用两个侦听器端口，通过 RabbitMQ Web 控制台或管理 API 访问代理。端口 443 提供标准的 HTTPS 访问权限，而端口 15671 是采用 TLS 加密的传统 RabbitMQ 管理端口。

属性

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

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 Amazon MQ for RabbitMQ 实施中，`X` 表示主要版本，`Y` 表示次要版本，`Z` 表示补丁版本号。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变化，则该版本变更被视为次要版本变更。例如，从版本 3 升级。 11 .28 到 3。 12 .13 被视为次要版本升级。

适用于 RabbitMQ 的亚马逊 MQ 建议所有经纪商使用支持的最新版本 RabbitMQ 4.2。有关如何升级代理引擎版本的说明，请参阅升级 Amazon MQ 代理引擎版本。

当您为 RabbitMQ 代理创建新的 Amazon MQ 时，您只需要指定主版本号和次要版本号即可。例如，RabbitMQ 4.2。如果您在创建代理时未指定引擎版本，Amazon MQ 会自动默认为最新的引擎版本。

> **Important**
>
> Amazon MQ 不支持数据流。创建流将导致数据丢失。
> 亚马逊 MQ 不支持在 JSON 中使用结构化日志。

亚马逊 MQ 支持 RabbitMQ 的两个主要版本版本：

- RabbitMQ 4

  亚马逊 MQ 仅在所有支持的实例大小的 mq.m7g 实例类型上支持 RabbitMQ 4 版本系列中的 RabbitMQ 4.2。
- RabbitMQ 3

  亚马逊 MQ 在 mq.t3、mq.m5 和 mq.m7g 实例类型上支持 RabbitMQ 3 版本系列中的 RabbitMQ 3.13，支持所有支持的实例大小。

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.m7g.large实例类型的结果，请使用和*engine-type*替RABBITMQ换为。*instance-type* mq.m7g.large

RabbitMQ 4

亚马逊 MQ 仅在所有支持的实例大小的 mq.m7g 实例类型上支持 RabbitMQ 4 版本系列中的 RabbitMQ 4.2。

亚马逊 MQ 支持从 RabbitMQ 3.13 到 RabbitMQ 4.2 的就地升级。有关更多信息，请参阅从 RabbitMQ 3 升级到 4。

> **Important**
>
> 亚马逊 MQ 上针对 RabbitMQ 4.2 经纪商的默认队列类型将是 “法定人数”。如果在创建队列期间未指定队列类型参数，则将创建法定队列。
> 出于耐久性需求，我们强烈建议在 RabbitMQ 4 上使用法定队列，因为不能保证经典队列在所有情况下都具有持久性。

亚马逊 MQ 上的 RabbitMQ 4 中引入了以下更改

- AMQP 1.0 作为核心协议：有关更多信息，请参阅协议。
- 本地铲子：除了 AMQP 0-9-1 和 AMQP 1.0 之外，Shovels 现在还支持一种名为 “本地” 的新协议。本地铲子内部基于 AMQP 1.0，但它们不是使用单独的 TCP 连接，而是使用集群节点之间的集群内连接和内部节点来发布和使用 APIs 消息。与 AMQP 0-9-1 和 AMQP 1.0 相比，它只能用于在同一集群内消费和发布，并且可以提供更高的吞吐量，同时使用的资源更少。
- 法定队列支持消息优先级：法定队列消息优先级始终处于活动状态，不需要策略即可起作用。一旦法定队列收到设置了优先级的消息，它就会启用优先级划分。法定人数队列内部仅支持两个优先级——高优先级和普通优先级。未设置优先级的邮件将映射为正常，优先级 0-4 也将映射为正常。优先级高于 4 的邮件将被映射到高。高优先级消息将优先于普通优先级消息，比例为 2:1，也就是说，对于每 2 条高优先级消息，队列将传送 1 条普通优先级消息（如果有）。因此，法定队列实现了一种非严格的 “公平份额” 优先级处理。这样可以确保在处理普通优先级消息时始终取得进展，但优先级较高的优先级优先级优先级为 2:1。
- Khepri：Khepri 被用作 RabbitMQ 4 经纪商的默认元数据存储
- 双向 TLS (mTLS)：亚马逊 MQ 支持 RabbitMQ 经纪商的双向 TLS (mTLS)，允许客户使用证书进行身份验证。有关更多信息，请参阅 mTLS 配置。
- SSL 证书身份验证插件：SSL 身份验证插件使用 mTLS 连接中的客户端证书对用户进行身份验证，允许使用 X.509 客户端证书而不是用户名和密码凭证进行身份验证。有关更多信息，请参阅 SSL 证书身份验证。
- HTTP 身份验证插件：HTTP 身份验证后端插件允许将身份验证和授权委托给外部 HTTP 服务。有关更多信息，请参阅 HTTP 身份验证和授权。
- JMS 支持：代理现在支持启用了 JMS 主题交换插件的 JMS 工作负载，允许 JMS 应用程序使用 RabbitMQ JMS 客户端进行连接。

亚马逊 MQ 上的 RabbitMQ 4 中已弃用以下功能

- 经典队列的镜像：继续支持经典队列，无需对客户端库和应用程序进行任何重大更改，但它们现在是一种非复制队列类型。客户端将能够连接到任何节点，以便从任何非复制的经典队列中发布和使用。为了复制和数据安全，建议使用法定队列。
- 移除全局 QoS：建议客户设置每个使用者 QoS（非全局），而不是全局 QoS，后者对整个频道使用单个共享预取。
- 对临时非排他性队列的支持：临时队列是指其生命周期与其声明所在节点的正常运行时间相关的队列。在单实例代理中，它们会在节点重新启动时被删除。在集群部署中，当它们所在的节点重新启动时，它们将被删除。我们建议使用队列 TTL 在闲置一段时间后自动删除未使用的空闲队列。继续支持独占队列，一旦移除所有与该队列的连接，该队列就会被删除。

在 Amazon MQ 上升级到 RabbitMQ 4.2 时，以下重大更改可能会影响您的应用程序

- 默认队列类型：RabbitMQ 4 代理上的默认队列类型设置为法定人数。如果在创建队列期间未指定队列类型参数，则将创建法定队列。
- 法定队列的默认重新传送限制设置为 20：重新传送 20 次或以上的邮件将被删除或丢弃（删除）。如果队列的常见情况是每封邮件 20 次传送，则必须为此类队列配置死写目标或更高的限制，以避免数据丢失。推荐的方法是通过政策。
- amqplib：低于 0.10.7 的 Node JS 客户端 amqplib 版本或任何 f rame_max < 8192 的 AMQP 客户端库都无法连接到 RabbitMQ
- 默认资源限制：适用于 RabbitMQ 的 Amazon MQ 对连接、频道、每个频道的使用者、队列、虚拟主机、铲子、交换和最大消息大小引入了默认资源使用限制。它们充当保护代理可用性的护栏，并且可以使用配置进行自定义，以满足您的特定要求。

亚马逊 MQ 上的 RabbitMQ 4 不支持以下功能

- 本地随机交换：Amazon MQ 不支持本地随机交换，因为 Amazon MQ 节点位于网络负载均衡器后面。
- 消息拦截器：亚马逊 MQ 不支持 RabbitMQ 消息拦截器。
- 每个队列指标：亚马逊 MQ 不会通过提供 RabbitMQ 4 经纪商的 RabbitMQ 队列指标。 AWS CloudWatchAmazon MQ 仍将通过以下方式提供经纪商级别的指标。 AWS CloudWatch您可以使用 RabbitMQ 管理 API 查询队列指标。我们建议以一分钟或更长的时间间隔查询特定队列的指标。

RabbitMQ 版本支持

下方的 Amazon MQ 版本支持日历显示了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。此次升级将在您的经纪商的预定维护时段内，即 end-of-support日期之后的45天内进行。

Amazon MQ 至少会在某个版本终止支持前 90 天发出通知。我们建议您在该 end-of-support日期之前升级您的经纪商，以免出现任何中断。此外，在支持终止日期后 30 天内，您不能在计划终止支持的版本上创建新的代理。

RabbitMQ 版本	Amazon MQ 终止支持
4.2（推荐使用）	
3.13	
3.12	2025 年 3 月 17 日

版本升级

您可以随时手动将代理升级到下一个受支持的主要版本或次要版本。有关手动升级代理的更多信息，请参阅升级 Amazon MQ 代理引擎版本。

Amazon MQ 管理所有使用 3.13 及以上版本的 RabbitMQ 代理升级到最新支持的补丁版本。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。

> **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：从 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。如果队列较长，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。为尽量减少影响，我们建议在流量较低的时段进行升级。有关版本升级最佳实践的更多信息，请参阅Amazon MQ for RabbitMQ 最佳实践。

将 RabbitMQ 的 Amazon MQ 3 代理升级到 4

亚马逊 MQ 支持从 RabbitMQ 3.13 到 RabbitMQ 4.2 的就地升级。就地升级不需要更改应用程序代码。在升级期间，Amazon MQ 会阻止与代理的所有连接。

Amazon MQ 不提供托管蓝绿色部署选项。如果您选择独立执行蓝绿部署，请参阅蓝绿部署。

> **Important**
>
> 升级之前，请查看 R abbitMQ 4 中引入的功能弃用、重大更改和新功能，以确保升级后操作顺畅。

下表比较了两种升级方法。

升级方法比较

考虑因素	就地升级（推荐）	蓝绿部署
停机时间	是的，在升级期间，Amazon MQ 会阻止与代理的所有连接。停机时间取决于队列深度。缩短排队时间可以减少停机时间。	不，您可以在不停机的情况下将生产者和消费者迁移到新的经纪商。
应用程序代码更改	无需更改。升级后，代理端点保持不变。	是的，您必须更新应用程序代码才能将生产者和消费者重定向到新的代理。

从 RabbitMQ 3 就地升级到 4

亚马逊 MQ 支持从 RabbitMQ 3.13 到 RabbitMQ 4.2 的主版本就地升级。只有所有支持的实例大小的`mq.m7g`实例类型都支持 RabbitMQ 4.2。

> **Note**
>
> 如果你的 Amazon MQ for RabbitMQ 3 代理启用了 Khepri，那么就没有到 RabbitMQ 4 的就地升级路径。有关更多信息，请参阅 RabbitMQ 版本可升级性。在这种情况下，请考虑蓝绿色部署。

> **Important**
>
> 升级持续时间取决于队列数量和队列深度。拥有大量队列和消息的经纪商将经历更长的停机时间。为了最大限度地减少停机时间，请尽量缩短排队时间。

步骤 1：升级代理实例类型

RabbitMQ 4.2 需要实例类型。`mq.m7g`如果您的代理已经在某种`mq.m7g`实例类型上运行，请移至the section called “步骤 2：将经典队列迁移到法定队列”。

使用 UpdateBrokerAPI 操作将代理的实例类型修改为`mq.m7g`。

使用 RebootBrokerAPI 重启代理以应用实例类型更改，或者等待下一个计划维护时段。

有关更多信息，请参阅升级 Amazon MQ 代理实例类型。

步骤 2：将经典队列迁移到法定队列

RabbitMQ 4 不支持经典镜像队列。如果代理有经典队列或经典镜像队列，Amazon MQ 将阻止原地升级到 RabbitMQ 4。

Amazon MQ 提供了一种队列迁移工具，用于将经典队列迁移到法定队列。此工具可通过 RabbitMQ Web 控制台的 “管理” > “队列迁移” 下或通过 HTTP API 进行访问。

要使用该工具，请参阅 Amazon MQ 队列迁移工具。

第 3 步：将引擎版本从 RabbitMQ 3.13 升级到 4.2

> **Note**
>
> 在升级期间，Amazon MQ 会阻止所有流向代理的外部流量。

> **Important**
>
> 如果您的 RabbitMQ 3.13 集群代理使用客户托管密钥 (CMK) 进行加密，则用于调用升级`UpdateBroker`到 4.2 版本的 IAM 角色必须对代理的加密密钥具有以下 AWS KMS 权限：
>
> - `kms:CreateGrant`

- `kms:DescribeKey`

如果调用角色没有这些权限，`UpdateBroker`API 会返回一个`403`错误，表明需要对 AWS KMS 密钥授予权限。要解决此错误，请在 IAM 角色的策略中添加`kms:CreateGrant`代理 AWS KMS 密钥 ARN 的`kms:DescribeKey`权限，然后重试升级。

使用 UpdateBrokerAPI 将 RabbitMQ 3.13 代理的待处理引擎版本设置为 4.2。

重新启动代理以应用更改，或者等待下一个预定的维护时段。

监控升级进度

您可以在 Amazon MQ 控制台上使用 DescribeBrokerAPI 或代理隔离状态来监控升级进度。

Amazon MQ 会在升级开始时进行升级资格检查。如果它识别了经典队列或者启用了 Khepri，则 Amazon MQ 会使用所需的操作代码将代理置于`CRITICAL_ACTION_REQUIRED`状态。`RABBITMQ_BROKER_NOT_UPGRADEABLE_TO_V4`Amazon MQ 不会应用主要版本升级，而是会让代理可供发布和使用。

要继续升级，请解决潜在的问题。有关更多信息，请参阅 RABBITMQ_BROKER_NOT_UPGRADEABLE_ TO_V4。

升级后更新资源限制配置

适用于 RabbitMQ 4 的 Amazon MQ 对连接、频道、每个频道的使用者、队列、虚拟主机、铲子、交换和最大消息大小引入了默认资源限制。在 RabbitMQ 3 代理上，这些资源配置了最大资源限制。就地升级到 RabbitMQ 4.2 后，亚马逊 MQ 会应用 RabbitMQ 4 的默认资源限制，该限制低于 RabbitMQ 3 中使用的最大资源限制。

Important

如果您的 RabbitMQ 3 代理使用的资源数量高于 RabbitMQ 4 默认限制，则该代理可能会拒绝升级后超过新限制的新连接、通道或队列声明。升级之前，请查看您的实例类型和部署模式的默认资源限制。升级完成后，更新代理配置以调整资源限制以满足您的工作负载要求。有关更多信息，请参阅资源限制配置。

从 RabbitMQ 3 到 4 的蓝绿色部署

亚马逊 MQ 不提供用于从 RabbitMQ 3.13 升级到 RabbitMQ 4.2 的托管蓝绿色部署选项。如果您选择独立执行蓝绿部署，则此方法需要更改应用程序代码才能将生产者和使用者重定向到新的代理。

有关详细说明，请参阅 RabbitM Q 文档中的蓝绿色部署。

Amazon MQ for RabbitMQ 代理的部署选项

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)

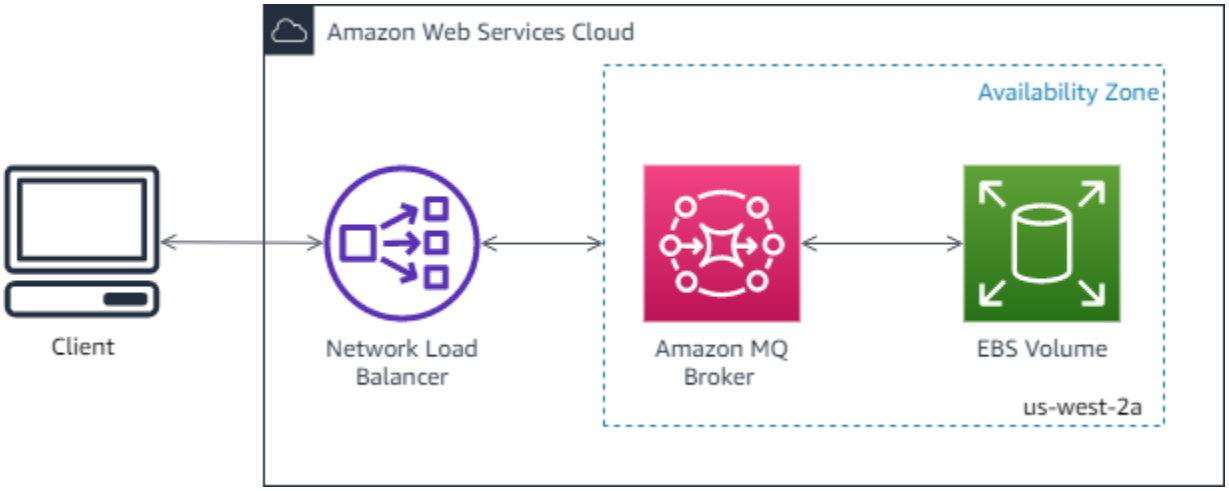
主题

选项 1：Amazon MQ for RabbitMQ 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。



选项 2：Amazon MQ for RabbitMQ 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。

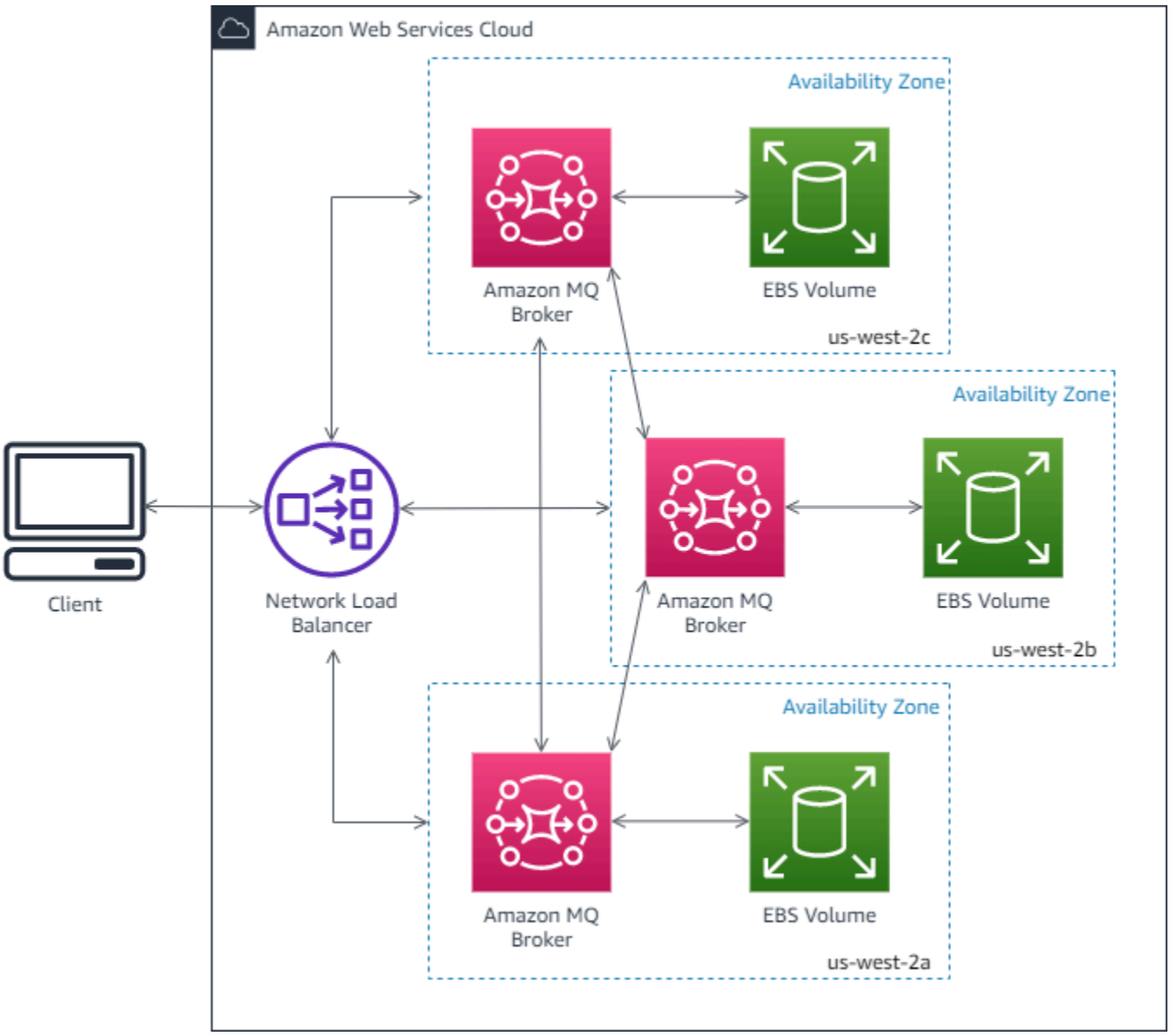
> **Note**
>
> 在集群部署中，如果发生可用区中断，Amazon MQ 将自动尝试将受影响的 RabbitMQ 节点重新放置到不同的可用区以维持集群大小。中断问题解决后，集群将在整个系统中自动重新平衡。 AZs

> **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 “步骤 1：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。如果您删除此策略，Amazon MQ 将自动重新创建此策略。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。

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 代理实例类型

代理实例类 (m7g) 和大小（大、中）的组合描述称为代理实例类型（例如 mq.m7g.large）。

对于集群和单实例部署，我们建议使用 mq.m7g 实例类型。

Amazon MQ 会在实例类型支持终止日期前至少提前 90 天发出通知。我们建议您在该 end-of-support 日期之前将您的代理升级到新的实例类型，以防止出现任何中断。

> **Important**
>
> 您不能将代理从mq.m7g或mq.m5实例类型降级为mq.t3.micro实例类型。该mq.t3.micro实例类型不支持集群部署。

m7g 集群部署的实例类型

我们建议在集群部署中使用 mq.m7g.x 实例类型。下表显示了可用于集群部署的 mq.m7g.x 实例类型。

实例类型	vCPU	内存（GiB）	网络基准/突发带宽 (Gbps)	推荐用途	仓储服务	每个节点的磁盘卷大小 (GB)
mq.m7g.medium	1	4	0.52/12.5	评估	EBS	5
mq.m7g.large	2	8	0.937 / 12.5	生产	EBS	15
mq.m7g.xlarge	4	16	1.876/12.5	生产	EBS	25
mq.m7g.2xlarge	8	32	3.75 / 15.0	生产	EBS	45
mq.m7g.4xlarge	16	64	7.5 / 15.0	生产	EBS	90

实例类型	vCPU	内存（GiB）	网络基准/突发带宽 (Gbps)	推荐用途	仓储服务	每个节点的磁盘卷大小 (GB)
mq.m7g.8xlarge	32	128	15Gb	生产	EBS	175
mq.m7g.12xlarge	48	192	22.5Gb	生产	EBS	260
mq.m7g.16xlarge	64	256	30Gb	生产	EBS	345

m7g 单实例部署的实例类型

下表显示了可用于单实例部署的 `mq.m7g.x` 实例类型。

实例类型	vCPU	内存（GiB）	网络基准/突发带宽 (Gbps)	推荐用途	仓储服务	每个节点的磁盘卷大小 (GB)
mq.m7g.medium	1	4	0.52/12.5	评估	EBS	200
mq.m7g.large	2	8	0.937 / 12.5	生产	EBS	200
mq.m7g.xlarge	4	16	1.876/12.5	生产	EBS	200
mq.m7g.2xlarge	8	32	3.75 / 15.0	生产	EBS	200
mq.m7g.4xlarge	16	64	7.5 / 15.0	生产	EBS	200
mq.m7g.8xlarge	32	128	15Gb	生产	EBS	200

实例类型	vCPU	内存（GiB）	网络基准/突发带宽 (Gbps)	推荐用途	仓储服务	每个节点的磁盘卷大小 (GB)
mq.m7g.12xlarge	48	192	22.5Gb	生产	EBS	200
mq.m7g.16xlarge	64	256	39 千兆位	生产	EBS	200

mq.m5 单实例部署的实例类型

下表显示了可用于单实例部署的 mq.m5.x 实例类型

实例类型	vCPU	内存（GiB）	网络基准/突发带宽 (Gbps)	推荐用途	仓储服务	每个节点的磁盘卷大小 (GB)
mq.t3.micro	2	1	0.064/5.0	评估	EBS	20
mq.m5.large	2	8	0.75/10.0	生产	EBS	200
mq.m5.xlarge	4	16	1.25 / 10.0	生产	EBS	200
mq.m5.2xlarge	8	32	2.5 / 10.0	生产	EBS	200
mq.m5.4xlarge	16	64	5.0 / 10.0	生产	EBS	200

mq.m5 集群部署的实例类型

下表显示了可用于集群部署的 mq.m5.x 实例类型

实例类型	vCPU	内存（GiB）	网络基准/突发带宽 (Gbps)	推荐用途	仓储服务	每个节点的磁盘卷大小 (GB)
mq.m5.large	2	8	0.75/10.0	生产	EBS	200
mq.m5.xlarge	4	16	1.25 / 10.0	生产	EBS	200
mq.m5.2xlarge	8	32	2.5 / 10.0	生产	EBS	200
mq.m5.4xlarge	16	64	5.0 / 10.0	生产	EBS	200

内存和磁盘警报

Amazon MQ 在每个 RabbitMQ 代理上配置内存和磁盘阈值，以防止资源耗尽。当超过阈值时，RabbitMQ 会触发警报并阻止发布者发送消息。使用不同连接的消费者可以继续正常运行。但是，如果发布者和消费者共享相同的连接，则该消费者也会被屏蔽。

> **Important**
>
> Amazon MQ 管理这些阈值，您无法对其进行修改。警报条件清除后，发布者将自动解除封锁。有关疑难解答信息，请参阅the section called “ RABBITMQ_MEMORY_ALARM ”和the section called “RABBITMQ_DISK_ALARM”。

记忆警报

该`vm_memory_high_watermark`参数定义了 RabbitMQ 代理在阻止发布者发送消息之前可以使用的最大内存量。当内存使用量超过此阈值时，RabbitMQ 会触发内存警报。有关更多信息，请参阅 RabbitMQ 网站上的内存警报。

对于`mq.m7g`实例类型，Amazon MQ 会设置以下绝对内存高水位线值：

实例类型	内存高水位线 (GiB)
mq.m7g.medium	1.8
mq.m7g.large	4.3
mq.m7g.xlarge	9.3
mq.m7g.2xlarge	19.3
mq.m7g.4xlarge	39.4
mq.m7g.8xlarge	79.7
mq.m7g.12xlarge	119.8
mq.m7g.16xlarge	160.1

对于mq.m5实例类型，Amazon MQ 将相对内存高水位线设置为 0.4（可用内存的 40%）。

mq.m7g实例上较高的内存阈值允许 RabbitMQ 在触发警报之前使用更多的可用内存。有关实例性能改进的更多信息，请参阅博客上的使用基于 AWS Graviton3 的 m7g mq.m7g 实例提高 Amazon MQ 上的 RabbitMQ 性能。 AWS

磁盘警报

该disk_free_limit参数定义了 RabbitMQ 节点所需的最小可用磁盘空间。当任何节点上的可用磁盘空间降至该限制以下时，RabbitMQ 会触发磁盘警报并阻止发布者发送消息。有关更多信息，请参阅 RabbitMQ 网站上的磁盘警报。

对于mq.m7g实例类型，Amazon MQ 设置了以下磁盘可用限制。单实例代理具有更高的磁盘可用限制以提供额外的保护，因为如果磁盘空间耗尽，它们就没有其他节点来处理流量。

部署模式	磁盘可用限制 (GiB)
单实例	10
Cluster	2

对于mq.m5实例类型，Amazon MQ 设置了以下磁盘可用限制。这些值适用于单实例部署和集群部署。

实例类型	磁盘可用限制 (GiB)
mq.m5.large	12
mq.m5.xlarge	20
mq.m5.2xlarge	36
mq.m5.4xlarge	69

由于mq.m7g实例的磁盘可用空间限制较低，因此与同等mq.m5实例相比，可用于消息存储的预配置磁盘容量更多。

Amazon MQ for RabbitMQ 大小调整指南

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

- 客户端和队列的数量
- 发送的消息量
- 保存在内存中的消息
- 冗余消息

建议仅m7g.medium在测试应用程序性能时使用较小的代理实例类型。我们建议使用更大的代理实例类型m7g.large以及高于或生产级别的客户端和队列、高吞吐量、内存中的消息和冗余消息。

> **Important**
>
> 您不能将代理从mq.m5或mq.m7g实例类型降级为mq.t3.micro实例类型。

请务必测试您的代理，以确定适合您的工作负载消息传递要求的实例类型和大小。

请务必使用 RabbitMQ 4 代理上的默认资源限制，根据 Amazon MQ 最佳实践为您的应用程序确定合适的实例大小。这些默认资源限制基于类型、m7g实例类型和法定队列。

- m7g 单实例部署的默认资源限制
- m7g 集群部署的默认资源限制

您可以将任何限制的值增加到实例类型和部署模式所定义的最大值。但是，我们强烈建议您在生产中使用之前使用增加的值来测试代理性能。

- m7g 单实例部署的最大资源限制
- m7g 集群部署的最大资源限制
- m5 单实例部署的最大资源限制
- m5 集群部署的最大资源限制
- 错误消息

> **Note**
>
> RabbitMQ 3.13 代理没有默认资源限制，但我们建议您使用建议的默认值。

默认资源限制

适用于 RabbitMQ 的 Amazon MQ 支持从 RabbitMQ 4 开始配置代理资源限制。当您创建代理时，Amazon MQ 会自动将默认值应用于这些资源限制。这些默认值充当保护您的经纪商可用性的护栏，同时适应常见的客户使用模式。您可以通过更改限制配置值来自定义代理行为，以更好地满足您的特定工作负载要求。

在进行更改之前，请注意：

> **Important**
>
> 1. 配置更改可能会影响代理的性能和可用性
> 2. 在更改任何默认配置选项之前，请先了解其影响
> 3. 首先在非生产环境中测试配置更改
> 4. 应用更改后监控代理运行状况

> **Important**
>
> 这些配置的默认值和支持的范围因 RabbitMQ 版本、实例类型和代理部署模式而异。

> **Important**
>
> 注意：将配置值超出支持范围的代理关联或创建代理将导致错误响应。

要了解如何为您的经纪商自定义这些默认资源限制，请参阅the section called “配置资源限制”。

适用于 RabbitMQ 4.2 代理的默认资源限制为

- m7g 单实例部署的默认资源限制
- m7g 集群部署的默认资源限制

默认资源限制

> **Important**
>
> 适用于 RabbitMQ 3 代理的 Amazon MQ，默认配置为最大资源限制，而 Amazon MQ 不提供覆盖资源限制配置的功能。

单实例代理的默认值

实例类型	每个节点的连接数	每个节点的通道数	每个通道的使用者数	队列	虚拟主机	Shovel	交易所	以字节为单位的邮件大小
mq.m7g.n dium	100	500	10	500	10	30	500	524288
mq.m7g.l rge	1500	4,500	10	1000	50	50	1000	524288

实例类型	每个节点的连接数	每个节点的通道数	每个通道的使用者数	队列	虚拟主机	Shovel	交易所	以字节为单位的邮件大小
mq.m7g.xarge	3000	9,000	10	2000	100	100	2000	524288
mq.m7g.2large	6000	18000	10	4,000	150	200	4,000	524288
mq.m7g.4large	12000	36,000	10	8000	200	400	8000	524288
mq.m7g.8large	24,000	72,000	10	16000	250	800	16000	524288
mq.m7g.1xlarge	36,000	108,000	10	24,000	300	1,200	24,000	524288
mq.m7g.1xlarge	48,000	144,000	10	32000	350	1,600	32000	524288

集群代理的默认值

实例类型	每个节点的连接数	每个节点的通道数	每个通道的使用者数	队列	虚拟主机	Shovel	交易所	以字节为单位的邮件大小
mq.m7g.medium	100	300	10	100	10	10	100	524288
mq.m7g.large	500	1500	10	1000	50	30	1000	524288
mq.m7g.xarge	1000	3000	10	2000	100	60	2000	524288

实例类型	每个节点的连接数	每个节点的通道数	每个通道的使用者数	队列	虚拟主机	Shovel	交易所	以字节为单位的邮件大小
mq.m7g.2large	2000	6000	10	4,000	150	120	4,000	524288
mq.m7g.4large	4000	12000	10	8000	200	240	8000	524288
mq.m7g.8large	8000	24,000	10	16000	250	480	16000	524288
mq.m7g.1xlarge	12000	36,000	10	24,000	300	720	24000	524288
mq.m7g.1xlarge	16000	48,000	10	32000	350	960	32000	524288

适用于 RabbitMQ 的亚马逊 MQ 资源上限上限

您可以配置资源限制，最高可达下表所示的最大值。要了解如何更新经纪商的资源限制，请参阅the section called “配置资源限制”。

单实例部署的 m7g 带法定队列的规模调整指南

下表列出了单实例代理的每种实例类型的最大限制值。

实例类型	Connections	渠道	每个通道的使用者数	队列	虚拟主机	Shovel	交易所	以字节为单位的邮件大小
mq.m7g.medium	300	900	1000	2,500	10	150	12500	134217728

实例类型	Connections	渠道	每个通道的使用者数	队列	虚拟主机	Shovel	交易所	以字节为单位的邮件大小
mq.m7g.large	5000	15000	1000	20000	1500	250	100000	134217728
mq.m7g.xlarge	10000	30000	1000	30000	1500	500	15万	134217728
mq.m7g.2xlarge	20000	60000	1000	40000	1500	1000	200,000	134217728
mq.m7g.4xlarge	40000	120,000	1000	60000	1500	2000	300,000	134217728
mq.m7g.8xlarge	80,000	240,000	1000	80,000	1500	4000	400,000	134217728
mq.m7g.12xlarge	120,000	360,000	1000	100000	1500	6000	500,000	134217728
mq.m7g.16xlarge	160000	480,000	1000	120,000	1500	8000	600,000	134217728

集群部署的 m7g 带法定队列的规模调整指南

下表列出了集群代理的每种实例类型的最大限制值。

实例类型	每个节点的连接数	每个节点的通道数	每个通道的使用者数	队列	虚拟主机	Shovel	交易所	以字节为单位的邮件大小
mq.m7g.medium	300	900	1000	500	10	50	500	134217728

实例类型	每个节点的连接数	每个节点的通道数	每个通道的使用者数	队列	虚拟主机	Shovel	交易所	以字节为单位的邮件大小
mq.m7g.la rge	5000	15000	1000	10000	1500	150	50000	134217728
mq.m7g.x arge	10000	30000	1000	15000	1500	300	75000	134217728
mq.m7g.2 large	20000	60000	1000	20000	1500	600	100000	134217728
mq.m7g.4 large	40000	120,000	1000	30000	1500	1200	15万	134217728
mq.m7g.8 large	80,000	240,000	1000	40000	1500	2400	200,000	134217728
mq.m7g.1 xlarge	120,000	360,000	1000	50000	1500	3600	250,000	134217728
mq.m7g.1 xlarge	160000	480,000	1000	60000	1500	4,800	300,000	134217728

M5 单实例部署的最大资源限制

下表列出了单实例代理的每种实例类型的最大限制值。

实例类型	Connections	渠道	每个通道的使用者数	队列	虚拟主机	Shovel
m5.large	5000	15000	1000	30000	1500	250
m5.xlarge	10000	30000	1000	60000	1500	500
m5.2xlarge	20000	60000	1000	120,000	1500	1000

实例类型	Connections	渠道	每个通道的使用者数	队列	虚拟主机	Shovel
m5.4xlarge	40000	120,000	1000	240,000	1000	2000

m5 集群部署的最大资源限制

下表列出了集群代理的每种实例类型的最大限制值。

实例类型	队列	每个通道的使用者数	Shovel
m5.large	10000	1000	150
m5.xlarge	15000	1000	300
m5.2xlarge	20000	1000	600
m5.4xlarge	30000	1000	1200

以下连接和通道限制按节点应用：

实例类型	Connections	渠道
m5.large	5000	15000
m5.xlarge	10000	30000
m5.2xlarge	20000	60000
m5.4xlarge	40000	120,000

集群代理的确切限制值可能低于指示值，具体取决于可用节点的数量以及 RabbitMQ 在可用节点之间分配资源的方式。如果超过限制值，则可以创建与其他节点的新连接并重试，也可以升级实例大小以增加最大限制

错误消息

超过限制时，将返回以下错误消息。所有值均基于 **`m7.large`** 单实例限制。

> **Note**
>
> 以下消息的错误代码可能会根据您使用的客户端库而变化。

Connection

```
ConnectionClosedByBroker 500 "NOT_ALLOWED - connection refused: node
connection limit (5000) is reached"
```

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 (15,000)"
```

使用者

```
ConnectionClosedByBroker: (530, 'NOT_ALLOWED - reached maximum (1,000) of
consumers per channel')
```

最大邮件大小

```
(406, 'PRECONDITION_FAILED - message size 524289 is larger than configured
max size 524288')
```

交易所

```
(406, "PRECONDITION_FAILED - cannot declare exchange 'limit_test_3' in
vhost '/': exchange limit of 10 is reached")
```

> **Note**
>
> 以下错误消息使用 HTTP 管理 API 格式。

队列

```
{"error":"bad_request","reason":"cannot declare queue 'my_queue': queue
limit in cluster (10,000) is reached"}]
```

Shovel

```
{"error":"bad_request","reason":"Validation failed\n\ncomponent shovel is
limited to 150 per node\n"}
```

Vhost

```
{"error":"bad_request","reason":"cannot create vhost 'my_vhost': vhost
limit of 1500 is reached"}
```

Amazon MQ for RabbitMQ 代理默认值

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。

亚马逊 MQ 为 RabbitMQ 3 和 RabbitMQ 4 的亚马逊 MQ 创建了不同的代理策略和虚拟主机限制。以下小节将详细讨论这些差异。

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：

适用于 RabbitMQ 的亚马逊 MQ 3：

- 单实例 – `AWS-DEFAULT-POLICY-SINGLE-INSTANCE`
- 集群部署 — `AWS-DEFAULT-POLICY-CLUSTER-MULTI-AZ` && `AWS-DEFAULT-QUORUM-QUEUES-POLICY-CLUSTER-MULTI-AZ`

适用于 RabbitMQ 的亚马逊 MQ 4：

- 单实例 – `AWS-DEFAULT-POLICY-SINGLE-INSTANCE`
- 集群部署 — `AWS-DEFAULT-POLICY-CLUSTER` && `AWS-DEFAULT-QUORUM-QUEUES-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。这些设置仅适用于 RabbitMQ 3 的亚马逊 MQ，不适用于 RabbitMQ 4。
> 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。

RabbitMQ 3 和 RabbitMQ 4 经纪人的设置

- **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**
>
> Cluster

- **`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**
>
> Cluster

RabbitMQ 3 的特定设置

- **`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

> **Important**
>
> `max-queuesmax-connections`并且仅适用于 RabbitMQ 3 的亚马逊 MQ。

> **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**
mq.m7g.medium	单实例	不适用	2,500	100
	Cluster	500,000	100	100
mq.m7g.large	单实例	不适用	20000	5000
	Cluster	8,000,000	10000	5000
mq.m7g.xlarge	单实例	不适用	30000	10000
	Cluster	90000,000	15000	10000
mq.m7g.2xlarge	单实例	不适用	40000	20000
	Cluster	10,000,000	40000	20000
mq.m7g.4xlarge	单实例	不适用	60000	40000
	Cluster	12,000,000	30000	40000
mq.m7g.8xlarge	单实例	不适用	80,000	80,000

实例类型	Deployment mode (部署模式)	max-length	max-queues	max-connections
	Cluster	20,000,000	40000	80,000
mq.m7g.12xlarge	单实例	不适用	100000	120,000
	Cluster	30,000,000	20000	120,000
mq.m7g.16xlarge	单实例	不适用	120,000	160000
	Cluster	40,000,000	50000	160000

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
m5.large	Cluster	8,000,000	10000	15000
m5.xlarge	单实例	不适用	30000	8000
m5.xlarge	Cluster	90000,000	10000	20000
m5.2xlarge	单实例	不适用	60000	15000
m5.2xlarge	Cluster	10,000,000	10000	40000
m5.4xlarge	单实例	不适用	15万	30000
m5.4xlarge	Cluster	12,000,000	10000	100000

配置 RabbitMQ 代理

配置包含 Cuttlefish 格式的 RabbitMQ 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。

属性

代理配置具有几个属性，例如：

- 名称 (MyConfiguration)
- 身份证（c-1234a5b6-78cd-901e-2fgh-3i45j6k178l9）
- 亚马逊资源名称 (ARN) (arn: aws: mq: us-east-2:123456789012: 配置:c-1234a5b678cd-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

主题

- 创建和应用 RabbitMQ 代理配置
- 编辑适用于 RabbitMQ 的亚马逊 MQ 配置修订版
- 亚马逊 MQ 上 RabbitMQ 的可配置值
- RabbitMQ 配置中支持 ARN

创建和应用 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

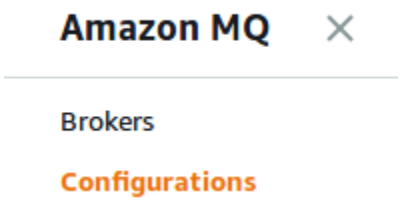
以下示例演示如何使用 AWS 管理控制台创建和应用 RabbitMQ 代理配置。

> **Important**
>
> 您只能使用 `DeleteConfiguration` API 删除配置。更多信息，请参阅 Amazon MQ API 参考中的配置。

创建新的配置

要将配置应用到代理，必须先创建配置。

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. 在创建配置页面的详细信息部分，输入配置名称（例如 `MyConfiguration`）并选择代理引擎版本。

   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

创建配置后，必须使用配置修订版编辑配置。

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。

*MyConfiguration*页面上会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。

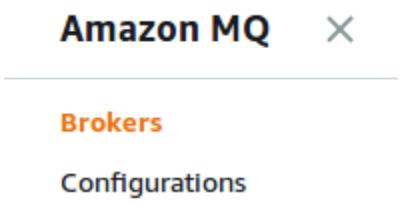
   将会保存配置的新修订。

> **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。
> 目前，您无法删除配置。

将配置修订应用到代理

创建配置修订版后，可以将配置修订版应用到代理。

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**
   >
   > 单实例代理在重启期间处于离线状态。对于集群代理，在代理重启期间，每次只有一个节点宕机。

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 Amazon MQ for RabbitMQ 配置修订版

以下说明介绍了如何编辑代理的配置修订版。

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。
> 目前，您无法删除配置。

可配置的值

您可以通过修改 AWS 管理控制台中的代理配置文件来设置以下代理配置选项的值。

除了下表中描述的值外，Amazon MQ 还支持与身份验证和授权以及资源限制相关的其他代理配置选项。有关这些配置选项的更多信息，请参见

- OAuth 2.0 配置
- LDAP 配置
- HTTP 配置
- SSL 配置
- mTLS 配置
- ARN 支持
- 资源限制
- AMQP 客户端 SSL 配置

配置	默认值	建议值	值	适用版本	说明
consumer_timeout	1800000 毫秒（30 分钟）	1800000 毫秒（30 分钟）	0 到 2,147,483,647 毫秒。Amazon MQ 还支持值	所有 版本	使用者不确认交付时，用于检测使用者交付确认的超时。

配置	默认值	建议值	值	适用版本	说明
			0，这意味着“无限”。		
heartbeat	60 秒	60 秒	60 到 3600 秒	所有 版本	定义 RabbitMQ 认为连接不可用之前的时间。
管理限制.operator_policy_changes.disabled	true	true	true，false	所有 版本	关闭更改操作员策略的功能。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。
quorum_queue.property_equeualence.relaxed_checks_on_redel	true	true	true，false	所有 版本	设置为 TRUE 时，应用程序在重新声明仲裁队列时将避免发生通道异常。
secure.management.http.headers.enabled	true	true	true，false	所有 版本	启用不可修改的 HTTP 安全标头。

配置使用者交付确认

您可以将 consumer_timeout 配置为检测消费者何时不确认交付。如果使用者没有在超时值内发送确认，通道将被关闭。例如，如果使用默认值 1800000 毫秒，若使用者在 1800000 毫秒内没有发送交付确认，通道将被关闭。Amazon MQ 还支持值 0，这意味着 “无限”。

配置检测信号

您可以配置检测信号超时，以了解连接何时中断或出现故障。检测信号值定义了连接被视为断开前的时间限制。

配置操作员策略

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 管理控制台 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。

配置队列声明的宽松检查

如果您已将经典队列迁移到法定队列但未更新客户端代码，则可以通过将 quorum_queue.property_equeue.property_equeance.relaxed_checks_on_redeclation 设置为 true 来避免在重新声明法定队列时出现频道异常。

配置 HTTP 安全标头

secure.management.http.headers.enabled 配置启用以下 HTTP 安全标头：

- X-Content-Type-Options: nosniff：防止浏览器执行内容嗅探，即用于推断网站文件格式的算法。
- X-Frame-Options: DENY：防止他人将管理插件嵌入自己网站上的框架来欺骗他人
- Strict-Transport-Security：max-age=47304000； includeSubDomains：强制浏览器在长时间（1.5 年）内与网站及其子域名建立后续连接时使用 HTTPS。

默认情况下，在 3.10 及更高版本上创建的 RabbitMQ 代理的 Amazon MQ 将
将 secure.management.http.headers.enabled 设置为 true。你可以通过将
secure.management.http.headers.enabled 设置为 true 来打开这些 HTTP 安全标头。如果你想选择退出这些 HTTP 安全标头，请将 secure.management.http.headers.enabled 设置为 false。

配置 OAuth 2.0 身份验证和授权

有关 OAuth 2.0 配置选项和为代理设置 OAuth 2.0 身份验证的信息，请参阅支持的 OAuth 2.0 配置和使用 OAuth 2.0 身份验证和授权。

配置 LDAP 身份验证和授权

有关 LDAP 配置选项和为代理设置 LDAP 身份验证的信息，请参阅支持的 LDAP 配置和使用 LDAP 身份验证和授权。

配置 HTTP 身份验证和授权

有关 HTTP 身份验证配置值和为代理设置 HTTP 身份验证的信息，请参阅 HTTP 身份验证和授权以及使用 HTTP 身份验证和授权。

> **Note**
>
> HTTP 身份验证插件仅适用于适用于 RabbitMQ 版本 4 及更高版本的亚马逊 MQ。

配置 SSL 证书身份验证

有关 SSL 证书身份验证配置值和为代理设置 SSL 证书身份验证的信息，请参阅 SSL 证书身份验证和使用 SSL 证书身份验证。

> **Note**
>
> SSL 证书身份验证插件仅适用于适用于 RabbitMQ 版本 4 及更高版本的亚马逊 MQ。

配置 mTLS

适用于 RabbitMQ 的 Amazon MQ 支持双向 TLS (mTLS)，以实现与各种终端节点和外部服务的安全连接。mTLS 要求客户端和服务器都使用证书进行身份验证，从而提高安全性。

> **Note**
>
> 只有适用于 RabbitMQ 版本 4 及更高版本的 Amazon MQ 才能使用私有证书颁发机构。

> **Important**
>
> 适用于 RabbitMQ 的 Amazon MQ 强制使用证书和私钥文件 AWS ARNs 。有关更多详细信息，请参阅 RabbitMQ 配置中的 ARN 支持。

本页内容

AMQP 终端节点

为与 AMQP 端点的客户端连接配置 mTLS。这与 SSL 证书身份验证一起使用。有关支持的配置，请参阅SSL 证书身份验证。

RabbitMQ 管理插件

配置 mTLS 以连接到 RabbitMQ 管理接口。

> **Note**
>
> 管理 API 不支持严格的 mTLS。

支持的配置

`aws.arns.management.ssl.cacertfile`

证书颁发机构文件，用于验证连接到管理接口的客户端证书。

management.ssl.verify

对等验证模式。支持的值：verify_none，verify_peer

management.ssl.depth

用于验证的最大证书链深度。

management.ssl.hostname_verification

主机名验证模式。支持的值：wildcard，none

不支持的 SSL 选项

不支持以下 SSL 配置值：

查看完整清单

- management.ssl.cert
- management.ssl.client_renegotiation
- management.ssl.dh
- management.ssl.dhfile
- management.ssl.fail_if_no_peer_cert
- management.ssl.honor_cipher_order
- management.ssl.honor_ecc_order
- management.ssl.key.RSAPrivateKey
- management.ssl.key.DSAPrivateKey
- management.ssl.key.PrivateKeyInfo
- management.ssl.log_alert
- management.ssl.password
- management.ssl.psk_identity
- management.ssl.reuse_sessions
- management.ssl.secure_renegotiate
- management.ssl.versions.$version
- management.ssl.sni

RabbitMQ 2.0 插件 OAuth

为从 Amazon MQ 到 2.0 身份提供商的连接配置 mTLS。 OAuth 有关支持的配置，请参阅OAuth 2.0 身份验证和授权。

RabbitMQ HTTP 身份验证插件

为从 Amazon MQ 到 HTTP 身份验证服务器的连接配置 mTLS。有关支持的配置，请参阅HTTP 身份验证和授权。

RabbitMQ LDAP 插件

为从亚马逊 MQ 到 LDAP 服务器的连接配置 mTLS。有关支持的配置，请参阅LDAP 身份验证和授权。

AMQP 客户端连接

为联合和 shovel 使用的 AMQP 客户端连接配置 TLS 对等验证。有关更多信息，请参阅 AMQP 客户端 SSL 配置。

> **Important**
>
> Amazon MQ 目前不支持为 AMQP 客户端连接配置客户端证书。因此，federation 和 shovel 无法连接到需要客户端证书身份验证的启用 MTLS 的代理。

资源限制配置

适用于 RabbitMQ 的亚马逊 MQ 支持从 RabbitMQ 4 开始配置代理资源限制。当您创建代理时，Amazon MQ 会自动将默认值应用于这些资源限制。这些默认值充当保护您的经纪商可用性的护栏，同时适应常见的客户使用模式。您可以通过更改限制配置值来自定义您的代理行为，以更好地满足您的特定工作负载要求。有关默认值和最大允许值的更多详细信息，请参阅the section called “大小调整指南”。

资源名称和配置密钥

资源名称	配置密钥
Connection	`connection_max`

资源名称	配置密钥
频道	channel_max_per_node
队列	cluster_queue_limit
Vhost	vhost_max
Shovel	runtime_parameters.limits.shovel
Exchange	cluster_exchange_limit
每个渠道的消费者	consumer_max_per_channel
最大消息大小	max_message_size

如何覆盖资源限制

您可以使用亚马逊 MQ API 和亚马逊 MQ 控制台来覆盖资源限制。

以下示例说明如何使用以下方法覆盖队列计数的默认限制 AWS CLI：

```
aws mq update-configuration --configuration-id <config-id> --data "$(echo
 "cluster_queue_limit=500" | base64 --wrap=0)"
```

成功调用会创建配置修订版。您必须将配置与您的 RabbitMQ 代理关联并重新启动代理才能应用覆盖。欲了解更多详情，请参阅 RabbitMQ Broker Configurations

配置中支持特定于实例的部分

在 RabbitMQ 4 中，亚马逊 MQ 支持配置数据中的章节。部分允许您在单个配置中定义特定于实例的资源限制。每个部分对应于特定的实例类型和部署模式组合。当您将配置与代理关联时，Amazon MQ 会自动为代理的实例类型和部署模式应用匹配部分。

> **Important**
>
> 分区支持仅在 RabbitMQ 4 上可用。如果您尝试将包含部分的配置应用于 RabbitMQ 3 代理，API 会返回。BadRequestException

章节语法

各节由双花括号分隔，格式如下：

```
{{<host-instance-family>.<size>.<mode>}}
```

该mode值表示部署模式：

- 1— 单实例代理
- 3— 集群代理

任何其他模式值均无效，并且 API 会返回错误。

以下示例显示了两种不同实例类型的配置数据，其中包含两个不同实例类型的部分：

```
connection_max = 1000

{{m7g.large.3}}
connection_max = 2000
{{m7g.large.3}}

{{m7g.xlarge.3}}
connection_max = 4000
{{m7g.xlarge.3}}
```

分区中允许的配置密钥

分区内仅支持以下资源限制配置密钥。在分区内添加任何其他配置密钥都会导致 API 错误。

- max_message_size
- channel_max_per_node
- connection_max
- cluster_queue_limit
- vhost_max
- consumer_max_per_channel

- runtime_parameters.limits.shovel
- cluster_exchange_limit

章节优先规则

当配置密钥同时出现在通用（顶级）部分和实例特定部分时，配置数据中稍后出现的值优先。例如，将以下配置应用于m7g.large集群代理将设置connection_max为2000：

```
connection_max = 1000

{{m7g.large.3}}
connection_max = 2000
{{m7g.large.3}}
```

反向顺序设置connection_max为1000，因为通用值排在最后：

```
{{m7g.large.3}}
connection_max = 2000
{{m7g.large.3}}

connection_max = 1000
```

> **Note**
>
> 如果配置数据未定义特定实例类型的值，则 Amazon MQ 将应用默认值。

示例

以下示例说明如何使用部分创建配置并使用将其与代理关联 AWS CLI。

使用分区更新配置

运行以下命令以针对多种实例类型更新具有特定实例资源限制的配置：

```
aws mq update-configuration \
```

```
    --configuration-id <config-id> \
    --data "$(echo -e "connection_max = 1000\nchannel_max_per_node =
64\n\n{{m7g.large.3}}\nconnection_max = 2000\nchannel_max_per_node =
128\n{{m7g.large.3}}\n\n{{m7g.xlarge.3}}\nconnection_max = 4000\nchannel_max_per_node
= 256\n{{m7g.xlarge.3}}" | base64 --wrap=0)"
```

此配置定义了以下值：

- 通用默认值：connection_max = 1000和 channel_max_per_node = 64
- m7g.large集群代理：connection_max = 2000和 channel_max_per_node = 128
- m7g.xlarge集群代理：connection_max = 4000和 channel_max_per_node = 256

将配置与代理关联

更新配置后，将其与您的代理关联并重新启动代理以应用更改。运行如下命令：

```
aws mq update-broker \
    --broker-id <broker-id> \
    --configuration id=<config-id>,revision=<revision-number>
```

资源限制覆盖错误

将配置值超出支持范围的代理关联或创建代理会导致类似于以下内容的错误响应：

```
Configuration Revision N for configuration:cluster_queue_limit has limit: of value:
 100000000 larger than maximum allowed limit:5000
```

有关按实例类型和部署模式划分的默认值和最大支持范围，请参阅the section called “默认资源限制”和the section called “最大资源限制”。

RabbitMQ 配置中支持 ARN

适用于 RabbitMQ 的亚马逊 MQ 支持某些 RabbitMQ 配置设置值的 AWS ARN。这是由 RabbitMQ 社区插件 rabbitmq-aws 启用的。该插件由亚马逊 MQ 开发和维护，也可以在非亚马逊 MQ 管理的自托管 RabbitMQ 代理中使用。

⚠ 重要注意事项

- aws 插件检索到的已解析的 ARN 值将在运行时直接传递给 RabbitMQ 进程。它们不会存储在 RabbitMQ 节点的其他地方。
- 适用于 RabbitMQ 的 Amazon MQ 需要一个可以由亚马逊 MQ 代入的 IAM 角色才能访问已配置的 ARN。这是通过设置进行配置的`aws.arns.assume_role_arn`。
- 使用包含 IAM 角色的代理配置调用 or CreateBroker 或 UpdateBroker API 的用户必须拥有该角色的`iam:PassRole`权限。
- IAM 角色必须与 RabbitMQ 代理存在于同一个 AWS 账户中。配置中的所有 ARN 都必须与 RabbitMQ 代理位于同一 AWS 区域。
- Amazon MQ 在担任 IAM 角色`aws:SourceArn`时会添加 IAM 全局条件密钥`aws:SourceAccount`。这些值必须用于附加到该角色的 IAM 策略中，以实现混乱的副手保护。

本页内容

支持的密钥

必需的 IAM 角色

`aws.arns.assume_role_arn`

Amazon MQ 为访问其他资源而扮演的 IAM 角色 ARN。 AWS 使用任何其他 ARN 配置时为必填项。

AMQP 终端节点

配置键	说明
aws.arns.ssl_options.cacertfile	用于 SSL/TLS 客户端连接的证书颁发机构文件。亚马逊 MQ 需要使用亚马逊 S3 或来存储证书。

RabbitMQ 管理插件

配置键	说明
aws.arns.management.ssl.cacertfile	管理接口 SSL/TLS 连接的证书颁发机构文件。亚马逊 MQ 需要使用亚马逊 S3 或来存储证书。

RabbitMQ OAuth 2.0 插件

配置键	说明
aws.arns.auth_oauth2.https.cacertfile	OAuth 2.0 HTTPS 连接的证书颁发机构文件。亚马逊 MQ 需要使用亚马逊 S3 或来存储证书。

RabbitMQ HTTP 身份验证插件

配置键	说明
aws.arns.auth_http.ssl_options.cacertfile	用于 HTTP 身份验证 SSL/TLS 连接的证书颁发机构文件。亚马逊 MQ 需要使用亚马逊 S3 或来存储证书。
aws.arns.auth_http.ssl_options.certfile	亚马逊 MQ 和 HTTP 身份验证服务器之间双向 TLS 连接的证书文件。亚马逊 MQ 需要使用亚马逊 S3 或来存储证书。

配置键	说明
aws.arns.auth_http.ssl_options.keyfile	用于 Amazon MQ 和 HTTP 身份验证服务器之间双向 TLS 连接的私钥文件。亚马逊 MQ 需要使用 AWS Secrets Manager 来存储私钥。

RabbitMQ LDAP 插件

配置键	说明
aws.arns.auth_ldap.ssl_options.cacertfile	LDAP SSL/TLS 连接的证书颁发机构文件。亚马逊 MQ 需要使用亚马逊 S3 或来存储证书。
aws.arns.auth_ldap.ssl_options.certfile	亚马逊 MQ 和 LDAP 服务器之间双向 TLS 连接的证书文件。亚马逊 MQ 需要使用亚马逊 S3 或来存储证书。
aws.arns.auth_ldap.ssl_options.keyfile	用于亚马逊 MQ 和 LDAP 服务器之间双向 TLS 连接的私钥文件。亚马逊 MQ 需要使用 AWS Secrets Manager 来存储私钥。
aws.arns.auth_ldap.dn_lookup_bind.password	LDAP DN 查找绑定的密码。Amazon MQ 要求使用 AWS Secrets Manager 将密码存储为纯文本值。
aws.arns.auth_ldap.other_bind.password	LDAP 其他绑定的密码。Amazon MQ 要求使用 AWS Secrets Manager 将密码存储为纯文本值。

IAM 政策示例

有关包括代入角色策略文档和角色策略文档在内的 IAM 策略示例，请参阅 CDK 示例实现。

使用 LDAP 身份验证和授权有关如何设置 AWS Secrets Manager 和 Amazon S3 资源的步骤，请参阅。

相关经纪人隔离状态

有关与 ARN 支持问题相关的经纪人隔离状态的信息，请参阅：

- RABBITMQ_INVALID_ASSUMEROLE
- RabbitMQ_INVALID_ARN_LDAP
- RABBITMQ_INVALID_ARN

示例方案

- 代理b-f0fc695e-2f9c-486b-845a-988023a3e55b已配置为使用 IAM 角色<role>访问 AWS Secrets Manager 密钥 <arn>
- 如果提供给 Amazon MQ 的角色没有 AWS Secrets Manager 密钥的读取权限，则 RabbitMQ 日志中将显示以下错误：

  ```
  [error] <0.254.0> aws_arn_config: {handle_assume_role,{error,{assume_role_failed,"AWS
   service is unavailable"}}}
  ```

  此外，经纪人将进入INVALID_ASSUMEROLE隔离状态。有关更多信息，请参阅 INVALID_ASSUMEROLE。
- LDAP 身份验证尝试将失败，并显示以下错误：

  ```
  [error] <0.254.0> LDAP bind failed: invalid_credentials
  ```
- 使用适当的权限修复 IAM 角色

AMQP 客户端 SSL 配置

Federation 和 shovel 使用 AMQP 在上游和下游代理之间进行通信。默认情况下，在 Amazon MQ for RabbitMQ 中为 AMQP 客户端启用 TLS 对等验证 4。使用此设置，在与上游代理建立连接时，在 Amazon MQ 代理上运行的联合和铲子 AMQP 客户端将执行同行验证。

在亚马逊 MQ 代理上运行的 AMQP 客户端支持与 Mozilla 相同的证书颁发机构。如果您不使用 ACM，请使用 M ozilla 包含的 CA 证书列表中由 CA 颁发的证书。如果您的本地代理使用其他证书颁发机构的证书，SSL 验证将失败。您可以针对这些用例禁用 TLS 对等验证。

Important

Amazon MQ 目前不支持为 AMQP 客户端连接配置客户端证书。因此，联合和 shovel 无法连接到需要客户端证书身份验证的启用 MTLS 的代理。

Important

在亚马逊 MQ for RabbitMQ 上，AMQP 客户端的 SSL 属性配置了 RabbitMQ 默认值（verify_none）。适用于 RabbitMQ 3 的亚马逊 MQ 不支持覆盖这些默认值。

Note

使用默认`verify_peer`设置，您可以在任意 2 个 Amazon MQ 经纪人之间建立联盟和切断连接，但这不支持在 Amazon MQ 经纪人与使用非 Amazon MQ CA 证书运行的私人经纪人或本地经纪人之间建立连接。要连接私有代理或本地代理，您需要在下游 Amazon MQ 代理上禁用对等验证。

AMQP 客户端 SSL 配置密钥

配置	配置密钥	支持的值
AMQP 客户端 SSL 对等验证	`amqp_client.ssl_options.verify`	`verify_none` ，`verify_peer`

如何覆盖 AMQP 客户端 SSL 对等验证

您可以在 RabbitMQ 4 代理上使用亚马逊 MQ API 和亚马逊 MQ 控制台覆盖 AMQP 客户端 SSL 对等验证。

以下示例说明如何使用以下方法覆盖 AMQP 客户端 SSL 对等验证： AWS CLI

```
aws mq update-configuration --configuration-id <config-id> --data "$(echo
 "amqp_client.ssl_options.verify=verify_none" | base64 --wrap=0)"
```

成功调用会创建配置修订版。您必须将配置与您的 RabbitMQ 代理关联并重新启动代理才能应用覆盖。欲了解更多详情，请参阅 Creating and applying broker configurations

> **Important**
>
> 使用时`verify_none`，SSL 加密仍处于活动状态，但未验证对等体的身份。仅在必要时才使用此设置，并确保您信任目标代理的网络路径。

适用于 RabbitMQ 身份验证和授权的亚马逊 MQ

Amazon MQ for RabbitMQ 支持以下认证与授权方法：

简单认证与授权

在此方法中，代理用户存储在 RabbitMQ 代理内部，并通过 Web 控制台或管理 API 进行管理。虚拟主机、交换机、队列和主题的权限直接在 RabbitMQ 中配置。这是默认方法。有关更多信息，请参阅简单身份验证和授权。

OAuth 2.0 认证与授权

在此方法中，代理用户及其权限由外部 OAuth 2.0 身份提供者（IdP）管理。虚拟主机、交换机、队列和主题的用户认证和资源权限通过 OAuth 2.0 提供程序的范围系统进行集中管理。这简化了用户管理，并实现了与现有身份系统的集成。有关更多信息，请参阅 OAuth 2.0 身份验证和授权。

IAM 身份验证和授权

在此方法中，代理用户通过 IAM 出站联合使用 AWS IAM 凭证进行身份验证。IAM 证书用于从 AWS 安全令牌服务 (STS) 获取 JWT 令牌，而这些 JWT 令牌则用作 OAuth 2.0 令牌进行身份验证。此方法利用了亚马逊 MQ 中对 RabbitMQ 的现有 OAuth 2.0 支持，RabbitMQ 充当 OAuth 2.0 身份提供商。 AWS 用户身份验证由 AWS IAM 处理，而虚拟主机、交易所、队列和主题的资源权限则通过 RabbitMQ 中配置的 IAM 策略和范围别名进行管理。有关更多信息，请参阅 IAM 身份验证和授权。

LDAP 身份验证和授权

在这种方法中，代理用户及其权限由外部 LDAP 目录服务管理。用户身份验证和资源权限通过 LDAP 服务器集中管理，允许用户使用其现有的目录服务凭据访问 RabbitMQ。有关更多信息，请参阅 LDAP 身份验证和授权。

HTTP 身份验证和授权

在这种方法中，代理用户及其权限由外部 HTTP 服务器管理。用户身份验证和资源权限通过 HTTP 服务器集中管理，允许用户使用自己的身份验证和授权提供程序访问 RabbitMQ。有关此方法的更多信息，请参阅 HTTP 身份验证和授权。

SSL 证书身份验证

亚马逊 MQ 支持 RabbitMQ 经纪商的双向 TLS (mTLS)。SSL 身份验证插件使用来自 mTLS 连接的客户端证书对用户进行身份验证。在这种方法中，使用 X.509 客户端证书而不是用户名和密码凭证对经纪人用户进行身份验证。根据受信任的证书颁发机构 (CA) 对客户端的证书进行验证，用户名是从证书的字段（例如公用名 (CN) 或主题备用名称 (SAN)）中提取的。此方法无需通过网络传输凭据即可提供强身份验证。有关更多信息，请参阅 SSL 证书身份验证。

> **Note**
>
> RabbitMQ 支持同时使用多种身份验证和授权方法。例如，您可以同时启用 OAuth 2.0 和简单（内部）身份验证。有关更多信息，请参阅 OAuth 2.0 教程中关于同时启用 OAuth 2.0 和简单（内部）身份验证的部分以及 Rabb itMQ 访问控制文档。

简单认证与授权

Amazon MQ for RabbitMQ 代理用户

> **Note**
>
> 本主题介绍如何使用 RabbitMQ 的默认内部身份验证和授权机制管理代理用户。有关所有支持的身份验证和授权方法的信息，请参阅适用于 RabbitMQ 身份验证和授权的 Amazon M Q。

每个 AMQP 0-9-1 客户端连接都有一个关联的用户。此用户必须经过身份验证。每个客户端连接还以虚拟主机 (vhost) 为目标。用户必须拥有该虚拟主机的一组权限。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。建立连接时，您可以指定用户凭据和目标虚拟主机。

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**
>
> Amazon MQ for RabbitMQ 不支持用户名“guest”，并会在您创建新代理时删除默认的访客账户。Amazon MQ 还将定期删除任何由客户创建的名为“guest”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。用新的登录*password*凭证替换*username*和。

```
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。
> - 如果您无法访问所有管理员账户，请参阅恢复代理访问权限以使用 IAM 身份验证进行恢复。

`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。

OAuth 适用于 RabbitMQ 的亚马逊 MQ 的 2.0 身份验证和授权

适用于 RabbitMQ 的亚马逊 MQ 支持多种身份验证和授权方法。有关所有支持方法的信息，请参阅适用于 RabbitMQ 代理的 Amazon MQ 身份验证和授权。

在 OAuth 2.0 身份验证和授权中，代理用户及其权限由外部 OAuth 2.0 身份提供商 (IdP) 管理。虚拟主机、交易所、队列和主题的用户身份验证和资源权限通过 OAuth 2.0 提供商的 scope 系统进行集中管理。这简化了用户管理，并实现了与现有身份系统的集成。

> **⚠ 重要注意事项**
>
> - OAuth Amazon MQ 不支持 ActiveMQ 经纪商的 2.0 集成。
> - Amazon MQ for RabbitMQ 不支持由私有 CA 颁发的服务器证书。
> - RabbitMQ OAuth 2.0 插件不支持令牌内省端点和不透明的访问令牌。它也不会执行令牌吊销检查。
> - 您必须包含 IAM 权限`mq:UpdateBrokerAccessConfiguration`，才能在现有代理上启用 OAuth 2.0。
> - Amazon MQ 会自动创建一个名为 `monitoring-AWS-OWNED-DO-NOT-DELETE` 的系统用户，该用户仅具有监控权限。即使在 OAuth 支持 2.0 的代理上，该用户也使用 RabbitMQ 的内部身份验证系统，并且只能访问环回接口。

有关如何为适用于 RabbitMQ 代理的 Amazon MQ 配置 OAuth 2.0 的信息，请参阅。使用 OAuth 2.0 认证与授权

本页内容

支持的 OAuth 2.0 配置

适用于 RabbitMQ 的 Amazon MQ 支持 RabbitMQ 2.0 插件中的所有可配置变量，但以下情况除外 OAuth ：

- `auth_oauth2.https.cacertfile`
- `auth_oauth2.oauth_providers.{id/index}.https.cacertfile`
- `management.oauth_client_secret`

  由于 Amazon MQ 不支持此密钥，因此我们不支持将 UAA 用作身份提供程序 (IdP)。
- `management.oauth_resource_servers.{id/index}.oauth_client_secret`
- `auth_oauth2.signing_keys.{id/index}`

OAuth 2.0 身份验证的其他验证

Amazon MQ 还对 2.0 身份验证强制执行以下额外验证： OAuth

- 一切都 URLs 需要从头开始`https://`。
- 支持的签名算法：`Ed25519`、`Ed25519ph`、`Ed448`、`Ed448ph`、`EdDSA`、`ES256K`、`ES256`、`ES384`、`ES512`、`HS256`、和 `RS512`。

适用于 RabbitMQ 的亚马逊 MQ 的 IAM 身份验证和授权

适用于 RabbitMQ 的亚马逊 MQ 支持多种身份验证和授权方法。有关所有支持方法的信息，请参阅适用于 RabbitMQ 代理的 Amazon MQ 身份验证和授权。

IAM 身份验证和授权允许代理用户通过 AWS IAM 出站联合使用 IAM 凭证进行身份验证。在此方法中，IAM 证书用于从 AWS 安全令牌服务 (STS) 获取 JWT 令牌。这些 JWT 令牌用作身份验证的 OAuth 2.0 令牌，利用了亚马逊 MQ 中对 RabbitMQ 的现有 OAuth 2.0 支持， AWS 其中充当 2.0 身份提供商。 OAuth AWS IAM 处理用户身份验证，而虚拟主机、交易所、队列和主题的资源权限则通过 RabbitMQ 中配置的 IAM 策略和范围别名进行管理。

⚠ 重要注意事项

- RabbitMQ 3.13、4.2 及更高版本支持 IAM 身份验证。Amazon MQ 不支持 ActiveMQ 经纪商。

- IAM 身份验证要求配置并可在您的 AWS 账户中使用 IAM 出站联合。
- 此方法建立在 Amazon MQ for RabbitMQ 中的现有 OAuth 2.0 基础架构之上，用作 2.0 身份 AWS 提供商。 OAuth
- Amazon MQ 会自动创建一个名为 `monitoring-AWS-OWNED-DO-NOT-DELETE` 的系统用户，该用户仅具有监控权限。即使在支持 IAM 的代理上，该用户也使用 RabbitMQ 的内部身份验证系统，并且只能访问环回接口。

本页内容

IAM 身份验证的工作原理

适用于 RabbitMQ 的 Amazon MQ 的 IAM 身份验证使用 IAM 出站联合身份验证来启用 IA AWS M 凭证向 RabbitMQ 代理进行身份验证。IAM 证书用于从 AWS 安全令牌服务 (STS) 获取 JWT 令牌，而这些 JWT 令牌可用作 OAuth 2.0 令牌，用于向 RabbitMQ 代理进行身份验证。

限制

适用于 RabbitMQ 的 Amazon MQ 的 IAM 身份验证存在以下限制：

- 作用域声明配置 — 您不能直接使用范围声明，因为 STS 中的 JWT 令牌是嵌套的。关键是`sts.amazonaws.com`，这需要在 RabbitMQ 配置中使用作用域别名将 IAM 角色映射到 RabbitMQ 权限。此限制还会阻止使用 IAM 策略进行完全授权，而是需要配置 RabbitMQ 才能进行授权。

有关如何为您的 Amazon MQ for RabbitMQ 代理配置 IAM 身份验证和授权的信息，请参阅。使用 IAM 身份验证和授权

适用于 RabbitMQ 的亚马逊 MQ 的 HTTP 身份验证和授权

适用于 RabbitMQ 的 Amazon MQ 支持使用外部 HTTP 服务器对代理用户进行身份验证和授权。有关其他支持的方法，请参阅适用于 RabbitMQ 代理的 Amazon MQ 身份验证和授权。

> **Note**
>
> HTTP 身份验证插件仅适用于适用于 RabbitMQ 版本 4 及更高版本的亚马逊 MQ。

> **⚠ 重要注意事项**
>
> - HTTP 服务器需要可通过公共互联网访问。适用于 RabbitMQ 的 Amazon MQ 可以配置为使用双向 TLS 向 HTTP 服务器进行身份验证。
> - 适用于 RabbitMQ 的 Amazon MQ 强制使用需要访问本地文件系统的设置。 AWS ARNs 有关更多详细信息，请参阅 RabbitMQ 配置中的 ARN 支持。
> - 您必须包含 IAM 权限`mq:UpdateBrokerAccessConfiguration`，才能对现有代理启用 HTTP 身份验证。
> - Amazon MQ 会自动创建一个名为 `monitoring-AWS-OWNED-DO-NOT-DELETE` 的系统用户，该用户仅具有监控权限。即使在支持 HTTP 的代理上，该用户也使用 RabbitMQ 的内部身份验证系统，并且只能访问环回接口。Amazon MQ 通过添加受保护的用户标签来防止删除该用户。

有关如何为您的 Amazon MQ for RabbitMQ 代理配置 HTTP 身份验证的信息，请参阅。使用 HTTP 身份验证和授权

本页内容

支持的 HTTP 配置

适用于 RabbitMQ 的 Amazon MQ 支持 Rabb itMQ HTTP 身份验证插件中的所有可配置变量，但以下需要的例外情况除外。 AWS ARNs有关 ARN 支持的详细信息，请参阅 RabbitMQ 配置中的 ARN 支持。

需要的配置 ARNs

`auth_http.ssl_options.cacertfile`

请改用 `aws.arns.auth_http.ssl_options.cacertfile`

auth_http.ssl_options.certfile

请改用 aws.arns.auth_http.ssl_options.certfile

auth_http.ssl_options.keyfile

请改用 aws.arns.auth_http.ssl_options.keyfile

不支持的 SSL 选项

也不支持以下 SSL 配置选项：

查看完整清单

- auth_http.ssl_options.cert
- auth_http.ssl_options.client_renegotiation
- auth_http.ssl_options.dh
- auth_http.ssl_options.dhfile
- auth_http.ssl_options.honor_cipher_order
- auth_http.ssl_options.honor_ecc_order
- auth_http.ssl_options.key.RSAPrivateKey
- auth_http.ssl_options.key.DSAPrivateKey
- auth_http.ssl_options.key.PrivateKeyInfo
- auth_http.ssl_options.log_alert
- auth_http.ssl_options.password
- auth_http.ssl_options.psk_identity
- auth_http.ssl_options.reuse_sessions
- auth_http.ssl_options.secure_renegotiate
- auth_http.ssl_options.versions.$version
- auth_http.ssl_options.sni
- auth_http.ssl_options.crl_check

亚马逊 MQ 中 HTTP 配置的其他验证

Amazon MQ 还对 HTTP 身份验证和授权强制执行以下额外验证：

- `auth_http.http_method`必须是`get`或 `post`
- 以下路径配置必须使用 HTTPS URLs：
  - `auth_http.user_path`
  - `auth_http.vhost_path`
  - `auth_http.resource_path`
  - `auth_http.topic_path`
- 如果有任何设置需要使用 AWS ARN，则`aws.arns.assume_role_arn`必须提供。

适用于 RabbitMQ 的亚马逊 MQ 的 SSL 证书身份验证

适用于 RabbitMQ 的 Amazon MQ 支持使用 X.509 客户端证书对经纪人用户进行身份验证。有关其他支持的方法，请参阅适用于 RabbitMQ 代理的 Amazon MQ 身份验证和授权。

> **Note**
>
> SSL 证书身份验证插件仅适用于适用于 RabbitMQ 版本 4 及更高版本的亚马逊 MQ。

> **⚠ 重要注意事项**
>
> - 客户证书必须由受信任的证书颁发机构 (CA) 签名。适用于 RabbitMQ 的亚马逊 MQ 会在身份验证期间验证证书链。
> - 适用于 RabbitMQ 的 Amazon MQ 强制使用证书相关设置（例如 CA 证书）和需要访问本地文件系统的设置。 AWS ARNs 有关更多详细信息，请参阅 RabbitMQ 配置中的 ARN 支持。
> - Amazon MQ 会自动创建一个名为 `monitoring-AWS-OWNED-DO-NOT-DELETE` 的系统用户，该用户仅具有监控权限。即使在支持 SSL 证书的代理上，该用户也使用 RabbitMQ 的内部身份验证系统，并且仅限于环回接口访问。Amazon MQ 通过添加受保护的用户标签来防止删除该用户。

有关如何为您的 Amazon MQ for RabbitMQ 代理配置 SSL 证书身份验证的信息，请参阅。使用 SSL 证书身份验证

本页内容

- 亚马逊 MQ 中 SSL 配置的其他验证

支持的 SSL 配置

适用于 RabbitMQ 的 Amazon MQ 支持 SSL/TLS 配置客户端连接。有关 ARN 支持的详细信息，请参阅 RabbitMQ 配置中的 ARN 支持。

需要的配置 ARNs

`ssl_options.cacertfile`

请改用 `aws.arns.ssl_options.cacertfile`

SSL 证书登录配置

以下配置控制如何从客户端证书中提取用户名：

`ssl_cert_login_from`

指定用于提取用户名的证书字段。支持的值：

- `distinguished_name`-使用完整的可分辨名称
- `common_name`-使用公用名 (CN) 字段
- `subject_alternative_name`或 `subject_alt_name`-使用主题备用名称

`ssl_cert_login_san_type`

使用主题备用名称时，指定 SAN 类型。支持的值：`dns`、`ip`、`email`、`uri`、`other_name`

`ssl_cert_login_san_index`

使用主题备用名称时，指定要使用的 SAN 条目的索引（从零开始）。必须是非负整数。

用于客户端连接的 SSL 选项

以下 SSL 选项适用于客户端连接：

`ssl_options.verify`

对等验证模式。支持的值：`verify_none`，`verify_peer`

`ssl_options.fail_if_no_peer_cert`

如果客户端不提供证书，是否拒绝连接。布尔值。

`ssl_options.depth`

用于验证的最大证书链深度。

`ssl_options.hostname_verification`

主机名验证模式。支持的值：`wildcard`，`none`

不支持的 SSL 选项

不支持以下 SSL 配置选项：

查看完整清单

- `ssl_options.cert`
- `ssl_options.client_renegotiation`
- `ssl_options.dh`
- `ssl_options.dhfile`
- `ssl_options.honor_cipher_order`
- `ssl_options.honor_ecc_order`
- `ssl_options.key.RSAPrivateKey`
- `ssl_options.key.DSAPrivateKey`
- `ssl_options.key.PrivateKeyInfo`
- `ssl_options.log_alert`
- `ssl_options.password`
- `ssl_options.psk_identity`
- `ssl_options.reuse_sessions`
- `ssl_options.secure_renegotiate`
- `ssl_options.versions.$version`
- `ssl_options.sni`
- `ssl_options.crl_check`

亚马逊 MQ 中 SSL 配置的其他验证

Amazon MQ 还对 SSL 证书身份验证强制执行以下额外验证：

- 如果有任何设置需要使用 AWS ARN，则`aws.arns.assume_role_arn`必须提供。

适用于 RabbitMQ 的 Amazon MQ 的 LDAP 身份验证和授权

适用于 RabbitMQ 的 Amazon MQ 支持使用外部 LDAP 服务器对代理用户进行身份验证和授权。有关其他支持的方法，请参阅适用于 RabbitMQ 代理的 Amazon MQ 身份验证和授权。

⚠ 重要注意事项

- LDAP 服务器需要可通过公共互联网进行访问。适用于 RabbitMQ 的 Amazon MQ 可以配置为使用双向 TLS 向 LDAP 服务器进行身份验证。
- 适用于 RabbitMQ 的 Amazon MQ 强制使用密码等敏感的 LDAP 设置以及需要访问本地文件系统的设置。 AWS ARNs 有关更多详细信息，请参阅 RabbitMQ 配置中的 ARN 支持。
- 您必须包含 IAM 权限`mq:UpdateBrokerAccessConfiguration`，才能在现有代理上启用 LDAP。
- Amazon MQ 会自动创建一个名为 `monitoring-AWS-OWNED-DO-NOT-DELETE` 的系统用户，该用户仅具有监控权限。即使在支持 LDAP 的代理上，该用户也使用 RabbitMQ 的内部身份验证系统，并且只能访问环回接口。Amazon MQ 通过添加受保护的用户标签来防止删除该用户。

有关如何为 RabbitMQ 代理配置 Amazon MQ 的 LDAP 的信息，请参阅。使用 LDAP 身份验证和授权

本页内容

支持的 LDAP 配置

适用于 RabbitMQ 的 Amazon MQ 支持 Rabb itMQ LDAP 插件中的所有可配置变量，但以下需要的例外情况除外。 AWS ARNs有关 ARN 支持的详细信息，请参阅 RabbitMQ 配置中的 ARN 支持。

需要的配置 ARNs

`auth_ldap.dn_lookup_bind.password`

请改用 `aws.arns.auth_ldap.dn_lookup_bind.password`

auth_ldap.other_bind.password

请改用 aws.arns.auth_ldap.other_bind.password

auth_ldap.ssl_options.cacertfile

请改用 aws.arns.auth_ldap.ssl_options.cacertfile

auth_ldap.ssl_options.certfile

请改用 aws.arns.auth_ldap.ssl_options.certfile

auth_ldap.ssl_options.keyfile

请改用 aws.arns.auth_ldap.ssl_options.keyfile

不支持的 SSL 选项

也不支持以下 SSL 配置选项：

查看完整清单

- auth_ldap.ssl_options.cert
- auth_ldap.ssl_options.client_renegotiation
- auth_ldap.ssl_options.dh
- auth_ldap.ssl_options.dhfile
- auth_ldap.ssl_options.honor_cipher_order
- auth_ldap.ssl_options.honor_ecc_order
- auth_ldap.ssl_options.key.RSAPrivateKey
- auth_ldap.ssl_options.key.DSAPrivateKey
- auth_ldap.ssl_options.key.PrivateKeyInfo
- auth_ldap.ssl_options.log_alert
- auth_ldap.ssl_options.password
- auth_ldap.ssl_options.psk_identity
- auth_ldap.ssl_options.reuse_sessions
- auth_ldap.ssl_options.secure_renegotiate

- `auth_ldap.ssl_options.versions.$version`
- `auth_ldap.ssl_options.sni`

亚马逊 MQ 中 LDAP 配置的其他验证

Amazon MQ 还对 LDAP 身份验证和授权强制执行以下额外验证：

- `auth_ldap.log`无法设置为 `network_unsafe`
- LDAP 服务器必须使用 LDAPS。要`auth_ldap.use_ssl`么`auth_ldap.use_starttls`必须明确启用
- 如果有任何设置需要使用 AWS ARN，则`aws.arns.assume_role_arn`必须提供。
- `auth_ldap.servers`必须是有效的 IP 地址或有效的 FQDN
- 以下密钥必须是有效的 LDAP 可分辨名称：
  - `auth_ldap.dn_lookup_base`
  - `auth_ldap.dn_lookup_bind.user_dn`
  - `auth_ldap.other_bind.user_dn`
  - `auth_ldap.group_lookup_base`

插件

适用于 RabbitMQ 的亚马逊 MQ 还支持以下插件。

- RabbitMQ 管理插件
- 铲子插件
- 联邦插件
- 一致的哈希交换插件
- OAuth 2 个插件
- LDAP插件
- HTTP 插件
- SSL 证书插件
- aws 插件
- JMS 话题交换插件

- Prometheus 插件

RabbitMQ 管理插件

适用于 RabbitMQ 的亚马逊 MQ 支持 RabbitMQ 管理插件，该插件为 RabbitMQ 网络控制台提供了基于 HTTP 的管理 API 以及基于浏览器的用户界面。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。

Shovel 插件

适用于 RabbitMQ 的 Amazon MQ 支持 R abbitMQ 铲子插件，该插件允许您将消息从一个代理商的队列和交易所转移到另一个经纪商。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。

> **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。
> Amazon MQ 不支持使用静态 shovel。

仅支持动态铲子。动态铲子是使用运行时参数配置的，可以通过客户端连接以编程方式随时启动和停止。例如，使用 RabbitMQ 管理 API，您可以向以下 API 端点创建 PUT 请求来配置动态铲子。在示例中，{vhost} 可以替换为代理虚拟主机的名称，{name} 可以替换为新的动态铲子的名称。

```
/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。以下示例在 src-queue 中指定的本地队列和 dest-queue 中定义的远程队列之间配置了动态铲子。同样，你可以使用 src-exchange 和 dest-exchange 参数在两个交换之间配置铲子。

```
{
"value": {
"src-protocol": "amqp091",
"src-uri": "amqp://localhost",
"src-queue": "source-queue-name",
"dest-protocol": "amqp091",
"dest-uri": "amqps://b-c8352341-ec91-4a78-ad9c-a43f23d325bb.mq.us-
west2.amazonaws.com:5671",
"dest-queue": "destination-queue-name"
```

```
}
}
```

联合插件

亚马逊 MQ 支持使用 Rabbit MQ 联合插件的联合交换和队列。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，您可以使用以下请求正文向/ api/parameters/federation-upstream/%2f/myupstream 创建一个 POST 请求。

  ```
  {"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，您可以使用以下请求正文向/api/policies/%2f/federate-me 创建 POST 请求。

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

  > **Note**
  >
  > 请求正文假设服务器上的交换以 amq 开头命名。使用正则表达式 ^amq\. 将确保所有名称以“amq” 开头的交易所启用联合。您的 RabbitMQ 服务器上的交换器可以不同的名称命名。

一致性哈希交换器插件

适用于 RabbitMQ 的亚马逊 MQ 支持 RabbitMQ 一致哈希交换类型插件。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。

OAuth 2.0 插件

适用于 RabbitMQ 的亚马逊 MQ 支持 2 身份验证后OAuth 端插件。此插件根据您的经纪人配置有条件地启用。启用后，此插件提供 OAuth 2.0 身份验证和授权，并集成到外部 OAuth 2.0 身份提供商，用于集中用户管理和访问控制。有关 OAuth 2.0 身份验证的更多信息，请参阅OAuth 2.0 身份验证和授权。

LDAP插件

适用于 RabbitMQ 的亚马逊 MQ 支持 LDAP 身份验证后端插件。此插件根据您的经纪人配置有条件地启用。启用后，此插件提供 LDAP 身份验证和授权，并集成到外部 LDAP 目录服务，用于集中用户身份验证和授权。有关 LDAP 身份验证的更多信息，请参阅LDAP 身份验证和授权。

HTTP 插件

适用于 RabbitMQ 的亚马逊 MQ 支持 HTTP 身份验证后端插件。此插件根据您的经纪人配置有条件地启用。启用后，此插件提供 HTTP 身份验证和授权，并集成到外部 HTTP 服务器，用于集中用户身份验证和授权。有关 HTTP 身份验证的更多信息，请参阅HTTP 身份验证和授权。

> **Note**
>
> HTTP 身份验证插件仅适用于适用于 RabbitMQ 版本 4 及更高版本的亚马逊 MQ。

SSL 证书插件

亚马逊 MQ 支持 RabbitMQ 经纪商的双向 TLS (mTLS)。SSL 身份验证插件使用来自 mTLS 连接的客户端证书对用户进行身份验证。此插件根据您的经纪人配置有条件地启用。启用后，它使用 X.509 客

户端证书提供基于证书的身份验证，以实现强身份验证，而无需通过网络传输凭据。有关 SSL 证书身份验证的更多信息，请参阅SSL 证书身份验证。

> **Note**
>
> SSL 证书身份验证插件仅适用于适用于 RabbitMQ 版本 4 及更高版本的亚马逊 MQ。

aws 插件

根据您的代理配置，亚马逊 MQ 有条件地为 RabbitMQ 启用 aws 插件。此社区插件由 Amazon MQ 开发和维护，可 AWS ARNs 在 RabbitMQ 配置设置中使用 AWS 服务安全地检索凭证和证书。有关 ARN 支持的更多信息，请参阅。ARN support in RabbitMQ configuration

JMS 话题交换插件

亚马逊 MQ 始终为 RabbitMQ 启用 JMS 主题交换插件。它与 RabbitMQ JMS 客户端配合使用，允许新的和现有的 JMS 应用程序连接到 RabbitMQ 版的 Amazon MQ。

> **Note**
>
> JMS Topic Exchange 插件仅适用于适用于 RabbitMQ 版本 4 及更高版本的亚马逊 MQ。它默认处于启用状态，但只有在使用 RabbitMQ JMS 客户端运行 JMS 工作负载时才会激活。

Prometheus 插件

Prometheus 插件内置于 RabbitMQ 中，并且始终在亚马逊 MQ 上为 RabbitMQ 经纪商启用。从 RabbitMQ 4.2 开始，Amazon MQ 与 RabbitMQ 的开源策略保持一致，即使用 Prometheus 来处理每个节点的指标，而不是使用管理插件进行长期监控。

该插件通过`/metrics`、`/metrics/detailed`和`/metrics/memory-breakdown`端点以 Prometheus 文本格式公开指标。您可以使用 Prometheus 或兼容的监控工具抓取这些端点，以构建仪表板并设置警报。

亚马逊 MQ 还将这些 Prometheus 指标的精选子集发布到。 CloudWatch有关 CloudWatch 指标的更多信息，请参阅适用于 RabbitMQ 经纪商的亚马逊 MQ 可用 CloudWatch 指标。

有关端点的详细信息、身份验证和抓取配置，请参阅访问 Prometheus 指标。

受支持的协议

您可以使用 RabbitMQ 支持的任何编程语言以及为以下任何协议规范启用 TLS 来访问您的 RabbitMQ 代理：

- AMQP (0-9-1)
- AMQP 1.0
- JMS 1.1
- JMS 2.0
- JMS 3.1

适用于 RabbitMQ 的亚马逊 MQ JMS 支持

现在，你可以使用 RabbitMQ JMS 客户端在亚马逊 MQ 上运行 RabbitMQ 4 的 JMS 1.1、2.0 和 3.1 工作负载。

RabbitMQ JMS 客户端

RabbitMQ JMS 客户端是一个开源 JMS 客户端库，你需要用它来将 JMS 应用程序连接到亚马逊 MQ RabbitMQ 代理。欲了解更多信息，请访问官方 GitHub 存储库。

支持 JMS 1.1、2.0 和 3.1 APIs

从适用于 RabbitMQ 4 的亚马逊 MQ 开始，该插件始终处于启用状态。`jms-topic-exchange`因此，你可以使用适用于 RabbitMQ 4 的 Amazon MQ 和 RabbitMQ JMS 客户端来处理 JMS 工作负载。支持 JMS 1.1 中 APIs 定义的所有 JMS，但以下情况除外：

- 不支持服务器 APIs 会话。
- APIs 不支持 XA 事务。
- 不支持 JMS 队列目标的 JMS 选择器。
- 不支持 JMS `NoLocal` 订阅属性。

支持 JMS 2.0 和 JMS 3.1 APIs 中所有新添加的内容，但以下情况除外：

- `JMSProducer.setDeliveryDelay`不支持 API。

要了解有关将您的 JMS 应用程序连接到 RabbitMQ 代理的 Amazon MQ 的更多信息，请参阅有关将您的 JM S 应用程序连接到 RabbitMQ 代理的亚马逊 M Q 的教程

身份验证和授权

支持本节中列出的所有身份验证和授权机制。使用 JMS 客户端连接到代理时使用的凭据与使用 AMQP Java 客户端连接 RabbitMQ 代理时使用的凭证相同。

与 RabbitMQ 上的 AMQP 队列的互操作性

您可以使用 RabbitMQ JMS 客户端向 AMQP 交易所发送 JMS 消息并使用来自 AMQP 队列的消息（此功能不支持 JMS 主题）。这使您可以互操作或将某些 JMS 工作负载迁移到 AMQP 工作负载。如需了解更多信息，请访问官方客户文档。

将策略应用于 Amazon MQ for RabbitMQ

您可以使用 Amazon MQ 推荐的默认值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

在 RabbitMQ 引擎版本 3.13 及以下版本的 Amazon MQ 上，当前的默认运营商政策是：

```
vhost name pattern apply-to definition priority/
 default_operator_policy_AWS_managed .* classic_queues {"ha-mode":"all","ha-
sync-mode":"automatic","queue-version":2} 0
```

在 4.0 及更高版本中，默认操作员政策已更改为：

```
vhost name pattern apply-to definition priority/
 default_operator_policy_AWS_managed .* classic_queues {"queue-version":2} 0
```

此更改是必需的，因为 RabbitMQ 4 不支持经典队列镜像和 HA 策略设置。
不能创建同时适用于经典镜像队列和仲裁队列的策略。如果希望只将策略应用于仲裁队列，必须将 `--apply-to` 设置为 `quorum_queues`。如果您使用经典镜像队列和法定队列，必须创建一个包含 `--apply-to:classic_queues` 的独立策略以及一个法定队列策略。

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Amazon MQ for RabbitMQ 代理用户。

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.m7g.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**
>
> 用您的管理员登录凭据替换*`username`*和*`password`*。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。*`policy-name`*替换为保单的名称。*`broker-endpoint`*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":{"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**
>
> 用您的管理员登录凭据替换*username*和*password*。根据代理*max-connections*的实例大小和部署模式，替换为 Amazon MQ 推荐值。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
```

RabbitMQ on Amazon MQ 的仲裁队列

仲裁队列是一种复制队列类型，由领导节点（主副本）和跟随节点（其他副本）组成。如果领导节点变得不可用，仲裁队列会使用 Raft 共识算法，以多数票选出一个新的领导节点，而之前的领导节点会被降级为同一集群中的跟随节点。其余的跟随节点继续像以前一样复制。由于每个节点都位于不同的可用区中，因此，如果一个节点暂时不可用，消息将通过另一个可用区新选出的领导副本继续传递。

仲裁队列可用于处理毒丸消息，当消息失败并被多次重新排队时，就会出现毒丸消息。

如果出现以下情况，则不应使用仲裁队列：

- 使用临时队列
- 排队等待时间长
- 优先考虑低延迟

要声明仲裁队列，请将标头 `x-queue-type` 设置为 `quorum`。

主题

从经典队列迁移到 Amazon MQ for RabbitMQ 上的仲裁队列

您可以通过在同一集群上创建新的虚拟主机或就地迁移的方式，将经典镜像队列迁移到 3.13 或更高版本的 Amazon MQ 代理上的仲裁队列。

选项 1：使用 Amazon MQ for RabbitMQ 队列迁移工具从经典镜像队列迁移到法定队列

Amazon MQ 提供了一种队列迁移工具，用于将经典队列迁移到法定队列。该工具可通过 RabbitMQ Web 控制台的 “管理” > “队列迁移” 下或通过 HTTP API 进行访问。

要使用该工具，请参阅 Amazon MQ 队列迁移工具。

选项 2：使用新的虚拟主机从经典镜像队列迁移到法定队列

您可以通过在同一集群上创建新的虚拟主机，将经典镜像队列迁移到 3.13 或更高版本的 Amazon MQ 代理上的仲裁队列。

1. 在现有集群中，创建一个新的虚拟主机（vhost），默认队列类型设为仲裁队列。
2. 使用经典镜像队列，从新的 vhost 创建 联合插件，URI 指向旧的 vhost。
3. 使用 `rabbitmqadmin`，将旧 vhost 中的定义导出到新文件中。必须对架构文件进行更改，使其与仲裁队列兼容。有关您需要对文件进行的更改的完整列表，请参阅 RabbitMQ 仲裁队列文档中的移动定义。对文件进行必要更改后，将定义重新导入到新的 vhost。

4. 在新的 vhost 中创建新策略。有关仲裁队列的 Amazon MQ 策略配置建议，请参阅Amazon MQ for RabbitMQ 仲裁队列的策略配置。然后，将之前创建的 Federation 从旧 vhost 启动到新 vhost。
5. 将使用者和生产者指向新的 vhost。
6. 配置 Shovel 插件以移动所有剩余的消息。队列为空后，删除 Shovel。

从经典镜像队列就地迁移到仲裁队列

您可以通过就地迁移的方式，将经典镜像队列迁移到 3.13 或更高版本的 Amazon MQ 代理上的仲裁队列。

1. 停止使用者和生产者。
2. 创建新的临时仲裁队列。
3. 配置 Shovel 插件，将旧的经典镜像队列中的任何消息移到新的临时仲裁队列。将所有消息移到临时仲裁队列后，删除 Shovel。
4. 删除源经典镜像队列。然后，用与源经典镜像队列相同的名称和绑定重新创建一个仲裁队列。
5. 创建新的 Shovel，将消息从临时仲裁队列移到新的仲裁队列。

Amazon MQ for RabbitMQ 仲裁队列的策略配置

您可以在 Amazon MQ 上为 RabbitMQ 代理的仲裁队列添加特定策略配置。

为仲裁队列创建策略时，必须执行以下操作：

- 移除所有以 ha 开头的策略属性，例如 `ha-mode`、`ha-params`、`ha-sync-mode`、`ha-sync-batch-size`、`ha-promote-on-shutdown` 和 `ha-promote-on-failure`。
- 删除 `queue-mode`。
- 当溢出设置为 `reject-publish-dlx` 时更改溢出

> **Important**
>
> Amazon MQ for RabbitMQ 会应用策略中的所有属性或不应用任何属性。不能创建同时适用于经典镜像队列和仲裁队列的策略。如果希望只将策略应用于仲裁队列，必须将 `--apply-to` 设置为 `quorum_queues`。如果使用经典镜像队列和仲裁队列，则必须使用 `--apply-to:classic_queues` 创建单独的策略以及仲裁队列策略。

您无需修改 AWS-DEFAULT 策略，因为它们会自动采用“应用于”参数中的新队列类型。有关 Amazon MQ for RabbitMQ 的默认策略的更多信息，请参阅 配置操作员策略。

Amazon MQ for RabbitMQ 仲裁队列的最佳实践

我们建议在处理仲裁队列时使用以下最佳实践来提高性能。

通过设置传送限制来处理毒丸消息

当消息失败并被多次重新传送时，就会出现毒丸消息。您可以使用 delivery-limit 策略参数设置消息传送限制，以丢弃多次重新传送的消息。如果消息的重新传送次数超过了传送限制允许的次数，RabbitMQ 就会丢弃并删除该消息。设置了传送限制时，消息会在队列头部附近重新排队。

仲裁队列的消息优先级

仲裁队列没有消息优先级。如果您需要消息优先级，则必须创建多个仲裁队列。有关使用多个仲裁队列确定消息优先级的更多信息，请参阅 RabbitMQ 文档中的 Message priority。

使用默认的复制因子

对于使用仲裁队列的集群代理，Amazon MQ for RabbitMQ 默认使用三（3）个节点的复制因子。如果您对 x-quorum-initial-group-size 进行更改，Amazon MQ 将再次默认复制因子为 3。

Amazon MQ for RabbitMQ 最佳实践

遵循这些生产就绪指南，以在使用 Amazon MQ for RabbitMQ 代理时最大化代理性能并优化消息吞吐效率。

> **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）。

主题

- Amazon MQ for RabbitMQ 网络弹性与监控最佳实践

Amazon MQ for RabbitMQ 中代理设置和连接管理的最佳实践

代理设置和连接管理是防止代理消息吞吐量、资源利用率及处理生产工作负载能力出现问题的第一步。当创建和配置 Amazon MQ for RabbitMQ 代理时，请完成以下最佳实践：选择适当的实例类型、有效管理连接以及配置消息预取，以最大化代理性能。

> **Important**
>
> Amazon MQ for RabbitMQ 不支持用户名“guest”，并会在您创建新代理时删除默认的访客账户。Amazon MQ 还将定期删除任何由客户创建的名为“guest”的账户。

步骤 1：使用集群部署

对于生产工作负载，我们建议使用集群部署而非单实例代理，以确保高可用性和消息弹性。集群部署消除了单点故障并提供更好的容错能力。

集群部署包含分布在三个可用区的三个 RabbitMQ 代理节点，可提供自动失效转移，并确保即使整个可用区不可用也能持续运行。Amazon MQ 会自动在所有节点间复制消息，以确保在节点故障或维护期间的可用性。

集群部署对生产环境至关重要，并受 Amazon MQ 服务级别协议支持。

更多信息，请参阅 Amazon MQ for RabbitMQ 中的集群部署。

步骤 2：选择正确的代理实例类型

代理实例类型的消息吞吐量取决于应用程序用例。`M7g.medium` 应仅用于测试应用程序性能。在生产环境中使用较大实例前先使用此较小实例可以提高应用程序性能。在实例类型 `m7g.large` 及更大实例上，您可以使用集群部署来实现高可用性和消息持久性。较大的代理实例类型可以处理生产级别的客户端和队列、高吞吐量、内存中的消息和冗余消息。

有关选择正确实例类型的更多信息，请参阅 Amazon MQ for RabbitMQ 中的规模调整指南。

步骤 3：使用法定队列

对于 3.13 及以上版本的 RabbitMQ 代理，法定队列配合集群部署应作为生产环境中复制队列类型的默认选择。法定队列是一种现代复制队列类型，具有高可靠性、高吞吐量和稳定延迟的特点。

法定队列使用 Raft 共识算法以提供更好的容错能力。当主节点不可用时，法定队列通过多数投票自动选举新主节点，确保消息传递以最小中断持续进行。由于每个节点位于不同的可用区，即使整个可用区暂时不可用，您的消息系统仍保持可用。

要声明法定队列，请在创建队列时将头参数 `x-queue-type` 设置为 `quorum`。

有关法定队列的更多信息（包括迁移策略和最佳实践），请参阅 Amazon MQ for RabbitMQ 中的法定队列。

步骤 4：使用多个通道

为避免连接中断，请在单个连接上使用多个通道。应用程序应避免 1:1 的连接与通道比率。我们建议每个进程使用一个连接，每个线程使用一个通道。避免过度使用通道，以防通道泄漏。

Amazon MQ for RabbitMQ 中消息持久性与可靠性的最佳实践

在将应用程序移至生产环境前，请完成以下防止消息丢失和资源过度利用的最佳实践。

步骤 1：使用持久化消息和持久化队列

持久化消息有助于在代理崩溃或重启时保护数据持久性。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。在使用 AMQP 0-9-1 时，您可以通过设置“2”传送模式将消息标记为持久消息。

```
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());
```

步骤 2：配置发布者确认和消费者投递确认

确认消息已发送到代理的过程称为发布者确认。发布者确认告知您的应用程序何时可靠地存储了消息。发布者确认还有助于控制存储到代理的消息速率。若无发布者确认，则无法确认消息是否成功处理，且代理可能会丢弃其无法处理的消息。

同样，当客户端应用程序向代理发回消息的交付和使用确认时，称为使用者交付确认。在使用 RabbitMQ 代理时，这两种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` 方法来启用确认。AMQP 0-9-1 客户端也可以通过发送 `confirm.select` 方法来配置发布者确认。

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。

您可以配置 consumer_timeout，以便在使用者不确认交付时进行检测。如果使用者没有在超时值内发送确认，通道将被关闭，您将收到 PRECONDITION_FAILED。要诊断错误，请使用 UpdateConfigurationAPI 增加consumer_timeout值。

步骤 3：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。

您还可以定义自动删除和 TTL 策略，以主动减少资源使用量，并尽量减少对消费者的侵 NACKs 害。在代理上重新排队消息会占用 CPU 密集型，因此大量消息可能会影响代理性能。 NACKs

Amazon MQ for RabbitMQ 中性能优化与效率的最佳实践

您可通过最大化吞吐量、最小化延迟及确保高效资源利用来优化 Amazon MQ for RabbitMQ 代理性能。请完成以下最佳实践以优化应用程序性能。

步骤 1：保持消息大小低于 1 MB

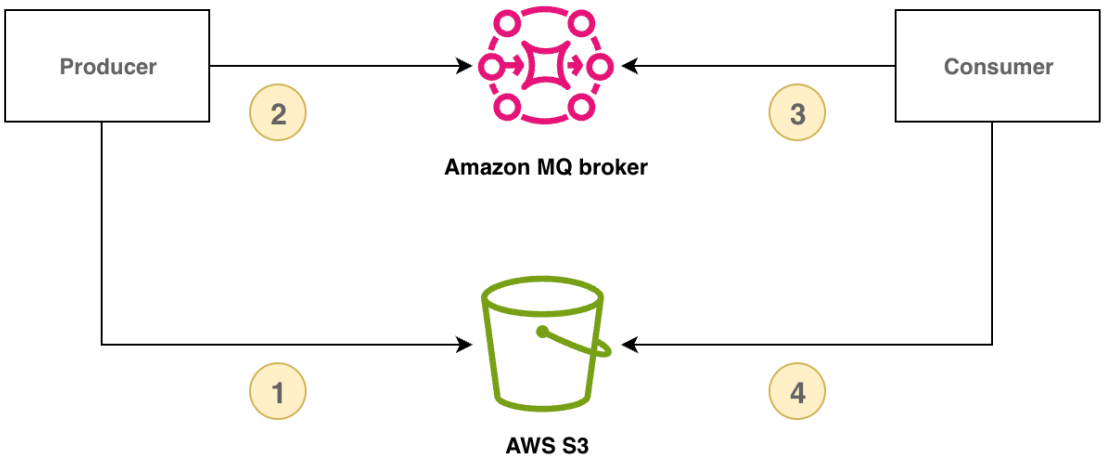
我们建议将消息保持在 1 兆字节（MB）以下以获得最佳性能和可靠性。

RabbitMQ 3.13 默认支持高达 128 MB 的消息大小，但大消息可能触发不可预测的内存警报，从而阻塞发布操作，并在跨节点复制消息时可能产生高内存压力。过大的消息还会影响代理重启和恢复过程，这会增加服务连续性的风险并可能导致性能下降。

使用认领检查模式存储和检索大有效载荷

为管理大消息，您可通过将消息有效载荷存储在外部存储中，并仅通过 RabbitMQ 发送有效载荷引用标识符来实现认领检查模式。消费者使用有效载荷引用标识符来检索和处理大消息。

下图演示了如何使用 Amazon MQ for RabbitMQ 和 Amazon S3 实现认领检查模式。



以下示例使用 Amazon MQ、AWS SDK for Java 2.x 和 Amazon S3 演示此模式：

1. 首先，定义一个用于存储 Amazon S3 引用标识符的 Message 类。

```
class Message {
    // Other data fields of the message...

    public String s3Key;
    public String s3Bucket;
}
```

2. 创建一个发布者方法，将有效载荷存储在 Amazon S3 中并通过 RabbitMQ 发送引用消息。

```
public void publishPayload() {
    // Store the payload in S3.
    String payload = PAYLOAD;
    String prefix = S3_KEY_PREFIX;
    String s3Key = prefix + "/" + UUID.randomUUID();
    s3Client.putObject(PutObjectRequest.builder()
        .bucket(S3_BUCKET).key(s3Key).build(),
        RequestBody.fromString(payload));

    // Send the reference through RabbitMQ.
    Message message = new Message();
    message.s3Key = s3Key;
```

```
    message.s3Bucket = S3_BUCKET;
    // Assign values to other fields in your message instance.

    publishMessage(message);
}
```

3. 实现一个消费者方法，从 Amazon S3 检索有效载荷、处理有效载荷并删除 Amazon S3 对象。

```
public void consumeMessage(Message message) {
    // Retrieve the payload from S3.
    String payload = s3Client.getObjectAsBytes(GetObjectRequest.builder()
        .bucket(message.s3Bucket).key(message.s3Key).build())
        .asUtf8String();

    // Process the complete message.
    processPayload(message, payload);

    // Delete the S3 object.
    s3Client.deleteObject(DeleteObjectRequest.builder()
        .bucket(message.s3Bucket).key(message.s3Key).build());
}
```

步骤 2：使用 **basic.consume** 和长生命周期消费者

使用 basic.consume 配合长生命周期消费者比使用 basic.get 轮询单个消息更高效。更多信息，请参阅轮询单个消息。

步骤 3：配置预取

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)`。

步骤 4：将 Celery 5.5 或更高版本与法定队列结合使用

Python Celery（一个分布式任务队列系统）在高任务负载时可能生成大量非关键消息。这种额外的代理活动可能触发 the section called “ `RABBITMQ_MEMORY_ALARM` ” 并导致代理不可用。为降低触发内存警报的风险，请执行以下操作：

对于所有 Celery 版本

1. 关闭 `task_create_missing_queues` 以减轻队列抖动。
2. 然后关闭 `worker_enable_remote_control` 以停止动态创建 `celery@...pidbox` 队列。这将减少代理上的队列抖动。

```
worker_enable_remote_control = false
```

3. 要进一步减少非关键消息活动，请在启动 Celery 应用程序时worker-send-task-events通过不包含-E或--task-events标记来关闭 Celery。
4. 使用以下参数启动 Celery 应用程序：

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

对于 Celery 5.5 及以上版本

1. 升级到 Celery 5.5 版本（支持法定队列的最低版本）或更高版本。要检查您使用的 Celery 版本，请使用 `celery --version`。有关法定队列的更多信息，请参阅 the section called “仲裁队列”。
2. 升级到 Celery 5.5 或更高版本后，将 `task_default_queue_type` 配置为 "quorum"。
3. 然后，您还必须在代理传输选项中开启发布确认：

```
broker_transport_options = {"confirm_publish": True}
```

Amazon MQ for RabbitMQ 网络弹性与监控最佳实践

网络弹性和监控代理指标对于维护可靠的消息应用程序至关重要。完成以下最佳实践以实施自动恢复机制和资源监控策略。

步骤 1：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 I/O 循环中抛出未处理的异常、检测到套接字读取操作超时或服务器错过心跳，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。

步骤 2：监控代理指标和警报

我们建议您定期监控您的 Amazon MQ for RabbitMQ 代理的CloudWatch 指标和警报，以便在潜在问题影响您的消息传递应用程序之前识别和解决这些问题。主动监控对于维护弹性消息应用程序和确保最佳性能至关重要。

适用于 RabbitMQ 的 Amazon MQ 向其发布指标 CloudWatch ，这些指标提供了对代理性能、资源利用率和消息流的见解。要监控的关键指标包括内存使用率和磁盘使用率。您可以设置CloudWatch 警报，以了解您的代理何时接近资源限制或出现性能下降的情况。

监控以下基本指标：

`RabbitMQMemUsed` 和 `RabbitMQMemLimit`

监控内存使用率以防止可能阻塞消息发布的内存警报。

`RabbitMQDiskFree` 和 `RabbitMQDiskFreeLimit`

监控磁盘使用率以避免可能导致代理故障的磁盘空间问题。

对于集群部署，还要监控节点特定指标以识别节点特定问题。

> **Note**
>
> 有关如何防止高内存警报的更多信息，请参阅处理和防止高内存警报。

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 Getting Started Guide》中的 RabbitMQ Tutorials。

主题

编辑代理首选项

您可以编辑您的经纪商首选项，例如使用启用或禁用 CloudWatch 日志 AWS 管理控制台。

编辑 RabbitMQ 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 “CloudWatch 日志” 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 经纪商，Amazon MQ 会自动使用服务关联角色 (SLR) 向其发布一般日志。 CloudWatch有关更多信息，请参阅 the section called “使用服务关联角色”。
> - 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。

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。本教程将指导您安装 Pika、声明队列、设置发布者以向代理的默认交易所发送消息，以及设置使用者以接收来自队列的消息。

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
```

Permissions

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。下表将必要的最低权限描述为正则表达式 (regexp) 模式。

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 Amazon MQ for RabbitMQ 代理用户。

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>**— 您创建适用于 RabbitMQ 的亚马逊 MQ 经纪商所在的 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!'
```

第三步：创建消费者并接收消息

要创建从队列中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()
```

与您在上一步中创建的发布者类似，BasicMessageReceiver它继承BasicPikaClient并实现了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 BasicMessageReceiver 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。

> **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
- 步骤 1：应用 ha-sync-batch-size 策略
- 步骤 2：重新启动队列同步

- 后续步骤
- 相关资源

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Amazon MQ for RabbitMQ 代理用户。

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 “数字”。

     > **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**
>
> 用您的经纪商管理员登录凭证替换*username*和*password*。您可能需要根据队列中不同步消息的数量和大小来调整和校准 `ha-sync-batch-size` (*100*) 的值。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 '{"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。在策略行中，您的队列应列出您创建的新策略的名称（例如 `batch-size-policy`）。
3. 要使用减小的批处理大小重新启动同步过程，请首先取消队列同步。然后重新启动队列同步。

> **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 指标来监控 RabbitMQ 节点使用的内存量。 CloudWatch `RabbitMQMemUsed`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，请参阅访问亚马逊 MQ 的 CloudWatch 指标和适用于 RabbitMQ 经纪商的亚马逊 MQ 可用 CloudWatch 指标。
- 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偏好的更多信息，请参阅 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https://docs.aws.amazon.com//amazon-mq/latest/developer-guide/rabbitmq-defaults.html。

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。

减少连接和通道的数量

连接到您的 Amazon MQ 代理上的 RabbitMQ 可以通过客户端应用程序关闭，或者通过使用 RabbitMQ Web 控制台手动关闭。要使用 RabbitMQ Web 控制台关闭连接，请执行以下操作：

1. 登录 AWS 管理控制台 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。当您关闭连接时，Amazon MQ 上的 RabbitMQ 将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。

步骤 2：将基于 JVM 的应用程序连接到代理

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 客户端库和开发人员工具。

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）Connect 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 管理控制台为 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 的更多信息、Lambda 支持的 Amazon MQ 事件源选项以及事件源映射错误，请参阅开发人员指南中的AWS Lambda 将 Lambda 与 Amazon MQ 结合使用。

为 Amazon MQ for RabbitMQ 使用 OAuth 2.0 认证与授权

本教程介绍如何使用 Amazon Cognito 作为 OAuth 2.0 提供程序，为您的 Amazon MQ for RabbitMQ 代理配置 OAuth 2.0 认证。

> **Note**
>
> Amazon Cognito 在中国（北京）和中国（宁夏）不可用。

> **Important**
>
> 本教程专门针对 Amazon Cognito，但您也可以使用其他身份提供程序 (IdP)。更多信息，请参阅 OAuth 2.0 认证示例。

本页内容

配置 OAuth 2.0 认证的先决条件

您可以通过部署 AWS CDK 堆栈 Amazon Cognito stack for RabbitMQ OAuth 2 plugin 来设置本教程所需的 Amazon Cognito 资源。如果您要手动设置 Amazon Cognito，请确保在 Amazon MQ for RabbitMQ 代理上配置 OAuth 2.0 之前满足以下先决条件：

设置 Amazon Cognito 的先决条件

- 通过创建用户池设置 Amazon Cognito 端点。要执行此操作，请参阅标题为如何在 Amazon Cognito 中使用 OAuth 2.0：了解不同的 OAuth 2.0 授权类型的博客文章。
- 在用户池中创建一个名为 `rabbitmq` 的资源服务器，并定义以下范围：`read:all`、`write:all`、`configure:all` 和 `tag:administrator`。这些范围将与 RabbitMQ 权限关联。

  有关创建资源服务器的信息，请参阅《Amazon Cognito 开发人员指南》中的为用户池定义资源服务器（AWS 管理控制台）。
- 创建以下应用程序客户端：
  - 用于 `Machine-to-Machine application` 类型用户池的应用程序客户端。这是一个具有客户端密钥的机密客户端，将用于 RabbitMQ AMQP 客户端。有关应用程序客户端及其创建的更多信息，请参阅应用程序客户端类型和创建应用程序客户端。
  - 用于 `Single-page application` 类型用户池的应用程序客户端。这是一个公共客户端，将用于使用户登录 RabbitMQ 管理控制台。您必须更新此应用程序客户端，以将您将在以下过程中创建的 Amazon MQ for RabbitMQ 代理的端点作为允许的回调 URL 包含在内。更多信息，请参阅使用 Amazon Cognito 控制台设置托管登录。

设置 Amazon MQ 的先决条件

- 可正常运行的 Docker 安装，用于运行验证 OAuth 2.0 设置是否成功的 bash 脚本。
- AWS CLI 版本 >= `2.28.23`，以便在代理创建过程中可选择是否添加用户名和密码。

使用 AWS CLI 配置与 Amazon Cognito 的 OAuth 2.0 认证

以下过程展示了如何使用 Amazon Cognito 作为身份提供程序 (IdP)，为您的 Amazon MQ for RabbitMQ 代理设置 OAuth 2.0 认证。此过程使用 AWS CLI 来创建和配置必要的资源。

在以下过程中，请确保将占位符值（例如 configurationID 和 Revision、*<c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca>* 和 *<2>*）替换为实际值。

1. 使用 create-configuration AWS CLI 命令创建新配置，如下例所示。

```
aws mq create-configuration \
  --name "rabbitmq-oauth2-config" \
  --engine-type "RABBITMQ" \
```

```
  --engine-version "3.13"
```

此命令返回类似于以下示例的响应。

```
{
    "Arn": "arn:aws:mq:us-west-2:123456789012:configuration:c-fa3390a5-7e01-4559-
ae0c-eb15b38b22ca",
    "AuthenticationStrategy": "simple",
    "Created": "2025-07-17T16:03:01.759943+00:00",
    "Id": "c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca",
    "LatestRevision": {
    "Created": "2025-07-17T16:03:01.759000+00:00",
    "Description": "Auto-generated default for rabbitmq-oauth2-config on RabbitMQ
 3.13",
    "Revision": 1
    },
    "Name": "rabbitmq-oauth2-config"
}
```

2. 创建一个名为 **rabbitmq.conf** 的配置文件以使用 OAuth 2.0 作为认证与授权方法，如下例所示。

```
auth_backends.1 = oauth2

# FIXME: Update this value with the token signing key URL of your Amazon Cognito
 user pool.
# If you used the AWS CDK stack to deploy Amazon Cognito, this is one of the stack
 outputs.
auth_oauth2.jwks_url = ${RabbitMqOAuth2TestStack.JwksUri}
auth_oauth2.resource_server_id = rabbitmq
# Amazon Cognito does not include an audience field in access tokens
auth_oauth2.verify_aud = false

# Amazon Cognito does not allow * in its custom scopes. Use aliases to translate
 between Amazon Cognito and RabbitMQ.
auth_oauth2.scope_prefix = rabbitmq/
auth_oauth2.scope_aliases.1.alias = rabbitmq/read:all
auth_oauth2.scope_aliases.1.scope = rabbitmq/read:*/*
auth_oauth2.scope_aliases.2.alias = rabbitmq/write:all
auth_oauth2.scope_aliases.2.scope = rabbitmq/write:*/*
auth_oauth2.scope_aliases.3.alias = rabbitmq/configure:all
auth_oauth2.scope_aliases.3.scope = rabbitmq/configure:*/*
```

```
# Allow OAuth 2.0 login for RabbitMQ management console
management.oauth_enabled = true
# FIXME: Update this value with the client ID of your public application client
management.oauth_client_id
 = ${RabbitMqOAuth2TestStack.ManagementConsoleAppClientId}
# FIXME: Update this value with the base JWKS URI (without /.well-known/jwks.json)
auth_oauth2.issuer = ${RabbitMqOAuth2TestStack.Issuer}
management.oauth_scopes = rabbitmq/tag:administrator
```

此配置使用范围别名将 Amazon Cognito 中定义的范围映射到 RabbitMQ 兼容范围。

3. 使用 update-configuration AWS CLI 命令更新配置，如下例所示。在此命令中，添加您在本过程步骤 1 的响应中收到的配置 ID。例如 **c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca**。

```
aws mq update-configuration \
  --configuration-id "<c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca>" \
  --data "$(cat rabbitmq.conf | base64 --wrap=0)"
```

此命令返回类似于以下示例的响应。

```
{
    "Arn": "arn:aws:mq:us-west-2:123456789012:configuration:c-b600ac8e-8183-4f74-
a713-983e59f30e3d",
    "Created": "2025-07-17T16:57:04.520931+00:00",
    "Id": "c-b600ac8e-8183-4f74-a713-983e59f30e3d",
    "LatestRevision": {
        "Created": "2025-07-17T16:57:39.172000+00:00",
        "Revision": 2
    },
    "Name": "rabbitmq-oauth2-config",
    "Warnings": []
}
```

4. 使用您在本过程步骤 2 中创建的 OAuth 2.0 配置创建代理。要执行此操作，请使用 create-broker AWS CLI 命令，如下例所示。在此命令中，分别提供您在步骤 1 和步骤 2 的响应中获得的配置 ID 和修订号。例如，**c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca** 和 **2**。

```
aws mq create-broker \
 --broker-name "rabbitmq-oauth2-broker" \
 --engine-type "RABBITMQ" \
 --engine-version "3.13" \
 --host-instance-type "mq.m7g.large" \
```

```
 --deployment-mode "CLUSTER_MULTI_AZ" \
 --logs '{"General": true}' \
 --publicly-accessible \
 --configuration '{"Id": "<c-fa3390a5-7e01-4559-ae0c-
eb15b38b22ca>","Revision": <2>}' \
```

此命令返回类似于以下示例的响应。

```
{
    "BrokerArn": "arn:aws:mq:us-west-2:123456789012:broker:rabbitmq-oauth2-
broker:b-2a1b5133-a10c-49d2-879b-8c176c34cf73",
    "BrokerId": "b-2a1b5133-a10c-49d2-879b-8c176c34cf73"
}
```

5. 使用 describe-broker AWS CLI 命令验证代理状态是否从 CREATION_IN_PROGRESS 转换为 RUNNING，如下例所示。在此命令中，提供您在上一步结果中获得的代理 ID，例如 **b-2a1b5133-a10c-49d2-879b-8c176c34cf73**。

```
aws mq describe-broker \
 --broker-id "<b-2a1b5133-a10c-49d2-879b-8c176c34cf73>"
```

此命令返回类似于以下示例的响应。以下响应是 describe-broker 命令返回的完整输出的缩写版本。此响应显示代理状态以及用于保护代理的认证策略。在此情况下，config_managed 认证策略表明代理使用 OAuth 2 认证方法。

```
{
    "AuthenticationStrategy": "config_managed",
    ...,
    "BrokerState": "RUNNING",
    ...
}
```

要使用 OAuth2 登录 RabbitMQ 管理控制台，需要将代理端点作为有效回调 URL 添加到相应的 Amazon Cognito 应用程序客户端中。更多信息，请参阅我们的示例 Amazon Cognito CDK 堆栈设置中的步骤 5。

6. 使用以下 perf-test.sh 脚本验证 OAuth 2.0 认证与授权。

使用此 bash 脚本测试与您的 Amazon MQ for RabbitMQ 代理的连接性。该脚本从 Amazon Cognito 获取令牌并验证连接是否配置正确。如果配置成功，您将看到代理发布和消费消息。

如果收到 ACCESS_REFUSED 错误，可以使用代理的 CloudWatch 日志对配置设置进行故障排除。您可以在 Amazon MQ 控制台中找到代理的 CloudWatch 日志组链接。

在此脚本中，您需要提供以下值：

- CLIENT_ID 和 CLIENT_SECRET：您可以在 Amazon Cognito 控制台的应用程序客户端页面找到这些值。
- Cognito 域：您可以在 Amazon Cognito 控制台上找到此值。在品牌定制下，选择域。在域页面，您可以在资源服务器部分找到此值。
- Amazon MQ 代理端点：您可以在 Amazon MQ 控制台的代理详细信息页面的连接下找到此值。

```
#! /bin/bash
set -e

# Client information
## FIXME: Update this value with the client ID and secret of your confidential
 application client
CLIENT_ID=${RabbitMqOAuth2TestStack.AmqpAppClientId}
CLIENT_SECRET=${RabbitMqOAuth2TestStack.AmqpAppClientSecret}

# FIXME: Update this value with the domain of your Amazon Cognito user pool
RESPONSE=$(curl -X POST ${RabbitMqOAuth2TestStack.TokenEndpoint} \
                -H "Content-Type: application/x-www-form-urlencoded" \
                -d
 "grant_type=client_credentials&client_id=${CLIENT_ID}&client_secret=${CLIENT_SECRET}&scope
configure:all rabbitmq/read:all rabbitmq/tag:administrator rabbitmq/write:all")


# Extract the access_token from the response.
# This token will be passed in the password field when connecting to the broker.
# Note that the username is left blank, the field is ignored by the plugin.
BROKER_PASSWORD=$(echo ${RESPONSE} | jq -r '.access_token')

# FIXME: Update this value with the endpoint of your broker. For
 example, b-89424106-7e0e-4abe-8e98-8de0dada7630.mq.us-east-1.on.aws.
BROKER_DNS=<broker_dns>
CONNECTION_STRING=amqps://:${BROKER_PASSWORD}@${BROKER_DNS}:5671

# Produce/consume messages using the above connection string
QUEUES_COUNT=1
```

```
PRODUCERS_COUNT=1
CONSUMERS_COUNT=1
PRODUCER_RATE=1

docker run -it --rm --ulimit nofile=40960:40960 pivotalrabbitmq/perf-test:latest \
    --queue-pattern 'test-queue-%d' --queue-pattern-from 1 --queue-pattern-to
 $QUEUES_COUNT \
    --producers $PRODUCERS_COUNT --consumers $CONSUMERS_COUNT \
    --id "test${QUEUES_COUNT}q${PRODUCERS_COUNT}p${CONSUMERS_COUNT}c
${PRODUCER_RATE}r" \
    --uri ${CONNECTION_STRING} \
    --flag persistent --rate $PRODUCER_RATE
```

配置与 Amazon Cognito 的 OAuth 2.0 和简单认证

当您创建使用 OAuth 2.0 认证的代理时，可以指定以下认证方法之一：

- 仅 OAuth 2.0：要使用此方法，在创建代理时请勿提供用户名和密码。先前的过程展示了如何仅使用 OAuth 2.0 认证方法。
- OAuth 2.0 和简单认证同时使用：要使用此方法，在创建代理时请提供用户名和密码。同时，将 `auth_backends.2 = internal` 添加到代理配置中，如下述过程所示。

在以下过程中，请确保将占位符值（例如 *<ConfigurationId>* 和 *<Revision>*）替换为实际值。

1. 要同时使用两种认证方法，请创建代理配置，如下例所示。

```
auth_backends.1 = oauth2
auth_backends.2 = internal

# FIXME: Update this value with the token signing key URL of your Amazon Cognito
 user pool
auth_oauth2.jwks_url = ${RabbitMqOAuth2TestStack.JwksUri}
auth_oauth2.resource_server_id = rabbitmq
auth_oauth2.verify_aud = false

auth_oauth2.scope_prefix = rabbitmq/
auth_oauth2.scope_aliases.1.alias = rabbitmq/read:all
auth_oauth2.scope_aliases.1.scope = rabbitmq/read:*/*
auth_oauth2.scope_aliases.2.alias = rabbitmq/write:all
auth_oauth2.scope_aliases.2.scope = rabbitmq/write:*/*
auth_oauth2.scope_aliases.3.alias = rabbitmq/configure:all
```

```
auth_oauth2.scope_aliases.3.scope = rabbitmq/configure:*/*
```

此配置使用范围别名将 Amazon Cognito 中定义的范围映射到 RabbitMQ 兼容范围。

2. 创建同时使用两种认证方法的代理，如下例所示。

```
aws mq create-broker \
 --broker-name "rabbitmq-oauth2-broker-with-internal-user" \
 --engine-type "RABBITMQ" \
 --engine-version "3.13" \
 --host-instance-type "mq.m7g.large" \
 --deployment-mode "CLUSTER_MULTI_AZ" \
 --logs '{"General": true}' \
 --publicly-accessible \
 --configuration '{"Id": "<ConfigurationId>","Revision": <Revision>}' \
 --users '[{"Username":"<myUser>","Password":"<myPassword11>"}]'
```

3. 按照 配置与 Amazon Cognito 的 OAuth 2.0 认证 过程中步骤 5 和 6 所述，验证代理状态和认证方法设置配置是否成功。

对适用于 RabbitMQ 的亚马逊 MQ 使用 IAM 身份验证和授权

以下过程演示如何为 Amazon MQ for RabbitMQ 代理启用 AWS IAM 身份验证和授权。启用 IAM 后，用户可以使用 AWS IAM 凭证进行身份验证，以访问 RabbitMQ 管理 API 并通过 AMQP 进行连接。有关 IAM 身份验证如何与适用于 RabbitMQ 的 Amazon MQ 配合使用的详细信息，请参阅。the section called “IAM 身份验证和授权”

先决条件

- AWS 拥有 Amazon MQ for RabbitMQ 经纪商的 AWS 账户的管理员证书
- 使用这些管理员凭据（使用 AWS CLI 配置文件或环境变量）配置的 shell 环境
- AWS 已安装并配置 CLI
- `jq`已安装命令行 JSON 处理器
- `curl`已安装命令行工具

使用配置 IAM 身份验证和授权 AWS CLI

1. 设置环境变量

为您的经纪商设置所需的环境变量：

```
export AWS_DEFAULT_REGION=<region>
export BROKER_ID=<broker-id>
```

2. 启用出站 JWT 令牌

为您的 AWS 账户启用出站 Web 联合身份验证：

```
ISSUER_IDENTIFIER=$(aws iam enable-outbound-web-identity-federation --query
 'IssuerIdentifier' --output text)
echo $ISSUER_IDENTIFIER
```

输出以以下格式显示您账户的唯一发行人标识符网址`https://<id>.tokens.sts.global.api.aws`。

3. 创建 IAM 策略文档

创建授予获取 Web 身份令牌权限的策略文档：

```
cat > policy.json << 'EOF'
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "sts:GetWebIdentityToken",
                "sts:TagGetWebIdentityToken"
            ],
            "Resource": "*"
        }
    ]
}
EOF
```

4. 创建信任策略

   检索您的来电者身份并创建信任策略文档：

```
CALLER_ARN=$(aws sts get-caller-identity --query Arn --output text)
cat > trust-policy.json << EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "$CALLER_ARN"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF
```

5. 创建 IAM 角色

   创建 IAM 角色并附加策略：

```
aws iam create-role --role-name RabbitMqAdminRole --assume-role-policy-document
 file://trust-policy.json
aws iam put-role-policy --role-name RabbitMqAdminRole --policy-name
 RabbitMqAdminRolePolicy --policy-document file://policy.json
```

6. 配置 RabbitMQ OAuth2 设置

   使用 OAuth2 身份验证和授权设置创建 RabbitMQ 配置文件：

```
cat > rabbitmq.conf << EOF
auth_backends.1 = oauth2
auth_backends.2 = internal
```

```
auth_oauth2.jwks_url = ${ISSUER_IDENTIFIER}/.well-known/jwks.json
auth_oauth2.resource_server_id = rabbitmq
auth_oauth2.scope_prefix = rabbitmq/

auth_oauth2.additional_scopes_key = sub
auth_oauth2.scope_aliases.1.alias = arn:aws:iam::$(aws sts get-caller-identity --
query Account --output text):role/RabbitMqAdminRole
auth_oauth2.scope_aliases.1.scope = rabbitmq/tag:administrator rabbitmq/read:*/*
 rabbitmq/write:*/* rabbitmq/configure:*/*
auth_oauth2.https.hostname_verification = wildcard

management.oauth_enabled = true
EOF
```

7. 更新代理配置

   将新配置应用于您的经纪商：

```
# Retrieve the configuration ID
CONFIG_ID=$(aws mq describe-broker --broker-id $BROKER_ID --query
 'Configurations[0].Id' --output text)

# Create a new configuration revision
REVISION=$(aws mq update-configuration --configuration-id $CONFIG_ID --data "$(cat
 rabbitmq.conf | base64 --wrap=0)" --query 'LatestRevision.Revision' --output text)

# Apply the configuration to the broker
aws mq update-broker --broker-id $BROKER_ID --configuration Id=$CONFIG_ID,Revision=
$REVISION

# Reboot the broker to apply changes
aws mq reboot-broker --broker-id $BROKER_ID
```

   等待代理状态恢复到RUNNING后再继续下一步。

8. 获取 JWT 代币

   担任 IAM 角色并获取 Web 身份令牌：

```
# Assume the RabbitMqAdminRole
ROLE_CREDS=$(aws sts assume-role --role-arn arn:aws:iam::$(aws sts get-caller-
identity --query Account --output text):role/RabbitMqAdminRole --role-session-name
 rabbitmq-session)

# Configure the session with temporary credentials
export AWS_ACCESS_KEY_ID=$(echo "$ROLE_CREDS" | jq -r '.Credentials.AccessKeyId')
export AWS_SECRET_ACCESS_KEY=$(echo "$ROLE_CREDS" | jq -r
 '.Credentials.SecretAccessKey')
export AWS_SESSION_TOKEN=$(echo "$ROLE_CREDS" | jq -r '.Credentials.SessionToken')

# Obtain the web identity token
TOKEN_RESPONSE=$(aws sts get-web-identity-token \
    --audience "rabbitmq" \
    --signing-algorithm ES384 \
    --duration-seconds 300 \
    --tags Key=scope,Value="rabbitmq/tag:administrator")

# Extract the token
TOKEN=$(echo "$TOKEN_RESPONSE" | jq -r '.WebIdentityToken')
```

9. 访问 RabbitMQ 管理 API

   使用 JWT 令牌访问 RabbitMQ 管理 API：

```
BROKER_URL=<broker-id>.mq.<region>.on.aws

curl -u ":$TOKEN" \
    -X GET https://${BROKER_URL}/api/overview \
    -H "Content-Type: application/json"
```

   成功的响应确认 IAM 身份验证工作正常。响应包含 JSON 格式的经纪商概述信息。

10. 使用 JWT 令牌通过 AMQP 进行连接

    使用带有性能测试工具的 JWT 令牌测试 AMQP 连接：

```
BROKER_DNS=<broker-endpoint>
CONNECTION_STRING=amqps://:${TOKEN}@${BROKER_DNS}:5671

docker run -it --rm --ulimit nofile=40960:40960 pivotalrabbitmq/perf-test:latest \
    --queue-pattern 'test-queue-%d' --queue-pattern-from 1 --queue-pattern-to 1 \
    --producers 1 --consumers 1 \
    --uri ${CONNECTION_STRING} \
    --flag persistent --rate 1
```

如果您收到ACCESS_REFUSED错误，则可以使用代理 CloudWatch 日志对配置设置进行故障排除。您可以在 Amazon MQ 控制台中找到您的代理的 CloudWatch 日志日志组链接。

对适用于 RabbitMQ 的 Amazon MQ 使用 LDAP 身份验证和授权

本教程介绍如何使用为 Amazon MQ 的 RabbitMQ 代理配置 LDAP 身份验证和授权。 AWS Managed Microsoft AD

本页内容

配置 LDAP 身份验证和授权的先决条件

您可以通过部署适用于 RabbitMQ 的 Amazon MQ 的AWS CDK 堆栈来设置本教程中所需的 AWS 资源。 AWS Managed Microsoft AD

此 CDK 堆栈会自动创建所有必要的 AWS 资源 AWS Managed Microsoft AD，包括 LDAP 用户和群组、Network Load Balancer、证书和 IAM 角色。有关堆栈创建的资源的完整列表，请参阅软件包自述文件。

如果您是手动设置资源而不是使用 CDK 堆栈，请确保在您的 Amazon MQ 上为 RabbitMQ 代理配置 LDAP 之前，请确保您有同等的基础架构。

设置 Amazon MQ 的先决条件

AWS CLI 版本 >= 2.28.23，使得在创建代理期间添加用户名和密码成为可选的。

使用 CLI 在 RabbitMQ 中配置 LDAP AWS

此过程使用 AWS CLI 来创建和配置必要的资源。在以下过程中，请确保将占位符值（例如 ConfigurationID 和 Revision 和）替换为它们的实际值。<c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca> <2>

1. 使用 `create-configuration` AWS CLI 命令创建新配置，如以下示例所示。

```
aws mq create-configuration \
  --name "rabbitmq-ldap-config" \
  --engine-type "RABBITMQ" \
  --engine-version "3.13"
```

此命令返回类似于以下示例的响应。

```
{
"Arn": "arn:aws:mq:us-west-2:123456789012:configuration:c-fa3390a5-7e01-4559-ae0c-
eb15b38b22ca",
    "AuthenticationStrategy": "simple",
    "Created": "2025-07-17T16:03:01.759943+00:00",
    "Id": "c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca",
    "LatestRevision": {
"Created": "2025-07-17T16:03:01.759000+00:00",
    "Description": "Auto-generated default for rabbitmq-ldap-config on RabbitMQ
 3.13",
    "Revision": 1
    },
    "Name": "rabbitmq-ldap-config"
}
```

2. 创建一个名为的配置文件`rabbitmq.conf`以使用 LDAP 作为身份验证和授权方法，如以下示例所示。将模板中的所有占位符值（标有`${RabbitMqLdapTestStack.*}`）替换为已部署的 AWS CDK 先决条件堆栈输出或等效基础架构中的实际值。

```
auth_backends.1 = ldap

# LDAP authentication settings - For more information,
```

```
# see https://www.rabbitmq.com/docs/ldap#basic

# FIXME: Replace the ${RabbitMqLdapTestStack.*} placeholders with actual values
# from your deployed prerequisite CDK stack outputs.
auth_ldap.servers.1 = ${RabbitMqLdapTestStack.NlbDnsName}
auth_ldap.dn_lookup_bind.user_dn = ${RabbitMqLdapTestStack.DnLookupUserDn}
auth_ldap.dn_lookup_base = ${RabbitMqLdapTestStack.DnLookupBase}
auth_ldap.dn_lookup_attribute = ${RabbitMqLdapTestStack.DnLookupAttribute}
auth_ldap.port = 636
auth_ldap.use_ssl = true
auth_ldap.ssl_options.verify = verify_peer
auth_ldap.log = network

# AWS integration for secure credential retrieval
# - see: https://github.com/amazon-mq/rabbitmq-aws
# The aws plugin allows RabbitMQ to securely retrieve credentials and certificates
# from AWS services.

# Replace the ${RabbitMqLdapTestStack.*} placeholders with actual ARN values
# from your deployed prerequisite CDK stack outputs.
aws.arns.auth_ldap.ssl_options.cacertfile = ${RabbitMqLdapTestStack.CaCertArn}
aws.arns.auth_ldap.dn_lookup_bind.password =
 ${RabbitMqLdapTestStack.DnLookupUserPasswordArn}
aws.arns.assume_role_arn = ${RabbitMqLdapTestStack.AmazonMqAssumeRoleArn}

# LDAP authorization queries - For more information,
# see: https://www.rabbitmq.com/docs/ldap#authorisation

# FIXME: Replace the ${RabbitMqLdapTestStack.*} placeholders with actual group DN
# values from your deployed prerequisite CDK stack outputs
# Uses Active Directory groups created by the prerequisite CDK stack
auth_ldap.queries.tags = '''
[{administrator, {in_group,
 "${RabbitMqLdapTestStack.RabbitMqAdministratorsGroupDn}"}},
{management,    {in_group,
 "${RabbitMqLdapTestStack.RabbitMqMonitoringUsersGroupDn}"}}]
'''

# FIXME: This provides all authenticated users access to all vhosts
# - update to restrict access as required
auth_ldap.queries.vhost_access = '''
{constant, true}
'''
```

```
# FIXME: This provides all authenticated users full access to all
# queues and exchanges - update to restrict access as required
auth_ldap.queries.resource_access = '''
{for, [    {permission, configure, {constant, true}},
     {permission, write,
      {for, [{resource, queue,    {constant, true}},
             {resource, exchange, {constant, true}}]}},
     {permission, read,
      {for, [{resource, exchange, {constant, true}},
             {resource, queue,    {constant, true}}]}}
    ]
}
'''

# FIXME: This provides all authenticated users access to all topics
# - update to restrict access as required
auth_ldap.queries.topic_access = '''
{for, [{permission, write, {constant, true}},
     {permission, read,  {constant, true}}
    ]
}
'''
```

3. 使用 update-configuration AWS CLI 命令更新配置，如以下示例所示。在此命令中，添加您在本过程步骤 1 的响应中收到的配置 ID。例如 c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca。

```
aws mq update-configuration \
  --configuration-id "<c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca>" \
  --data "$(cat rabbitmq.conf | base64 --wrap=0)"
```

此命令返回类似于以下示例的响应。

```
{
    "Arn": "arn:aws:mq:us-west-2:123456789012:configuration:c-b600ac8e-8183-4f74-
a713-983e59f30e3d",
    "Created": "2025-07-17T16:57:04.520931+00:00",
    "Id": "c-b600ac8e-8183-4f74-a713-983e59f30e3d",
```

```
    "LatestRevision": {
        "Created": "2025-07-17T16:57:39.172000+00:00",
        "Revision": 2
    },
    "Name": "rabbitmq-ldap-config",
    "Warnings": []
}
```

4. 使用您在本过程的步骤 2 中创建的 LDAP 配置创建代理。为此，请使用 `create-broker` AWS CLI 命令，如以下示例所示。在此命令中，分别提供您在步骤 1 和步骤 2 的响应中获得的配置 ID 和修订号。例如，`c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca` 和 2。

```
aws mq create-broker \
 --broker-name "rabbitmq-ldap-test-1" \
 --engine-type "RABBITMQ" \
 --engine-version "3.13" \
 --host-instance-type "mq.m7g.large" \
 --deployment-mode "CLUSTER_MULTI_AZ" \
 --logs '{"General": true}' \
 --publicly-accessible \
 --configuration '{"Id": "<c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca>","Revision":
 <2>}'
```

此命令返回类似于以下示例的响应。

```
{
    "BrokerArn": "arn:aws:mq:us-west-2:123456789012:broker:rabbitmq-ldap-
broker:b-2a1b5133-a10c-49d2-879b-8c176c34cf73",
    "BrokerId": "b-2a1b5133-a10c-49d2-879b-8c176c34cf73"
}
```

> **⚠ 经纪人命名限制**
>
> 由先决条件 CDK 堆栈创建的 IAM 角色将代理名称限制为开头。`rabbitmq-ldap-test`请确保您的代理名称遵循此模式，否则 IAM 角色将无权担任 ARN 解析的角色。

5. 使用 describe-broker AWS CLI 命令验证代理的状态是否从CREATION_IN_PROGRESS转换为RUNNING，如以下示例所示。在此命令中，提供您在上一步结果中获得的代理 ID，例如 b-2a1b5133-a10c-49d2-879b-8c176c34cf73。

```
aws mq describe-broker \
 --broker-id "<b-2a1b5133-a10c-49d2-879b-8c176c34cf73>"
```

此命令返回类似于以下示例的响应。以下响应是 describe-broker 命令返回的完整输出的缩写版本。此响应显示代理状态以及用于保护代理的认证策略。在这种情况下，config_managed身份验证策略表明代理使用 LDAP 身份验证方法。

```
{
"AuthenticationStrategy": "config_managed",
    ...,
    "BrokerState": "RUNNING",
    ...
}
```

6. 使用先决条件 CDK 堆栈创建的测试用户之一验证 RabbitMQ 访问权限

```
# FIXME: Replace ${RabbitMqLdapTestStack.ConsoleUserPasswordArn} with the actual
 ARN from your deployed prerequisite CDK stack outputs
CONSOLE_PASSWORD=$(aws secretsmanager get-secret-value \
  --secret-id ${RabbitMqLdapTestStack.ConsoleUserPasswordArn} \
  --query 'SecretString' --output text)

# FIXME: Replace BrokerConsoleURL with the actual ConsoleURL retrieved by
# calling describe-broker for the broker created above
# Call management API /api/overview (should succeed)
curl -u RabbitMqConsoleUser:$CONSOLE_PASSWORD \
  https://${BrokerConsoleURL}/api/overview

# Try to create a user (should fail - console user only has monitoring permissions)
curl -u RabbitMqConsoleUser:$CONSOLE_PASSWORD \
  -X PUT https://${BrokerConsoleURL}/api/users/testuser \
  -H "Content-Type: application/json" \
  -d '{"password":"testpass","tags":"management"}'
```

对适用于 RabbitMQ 的亚马逊 MQ 使用 HTTP 身份验证和授权

本教程介绍如何使用外部 HTTP 服务器为 Amazon MQ for RabbitMQ 代理配置 HTTP 身份验证和授权。

> **Note**
>
> HTTP 身份验证插件仅适用于适用于 RabbitMQ 版本 4 及更高版本的亚马逊 MQ。

本页内容

配置 HTTP 身份验证和授权的先决条件

您可以通过部署适用于 RabbitMQ 的 Amazon MQ 的AWS CDK 堆栈来设置本教程中所需的 AWS 资源 HTTP 身份验证集成。

此 CDK 堆栈会自动创建所有必要的 AWS 资源，包括 HTTP 身份验证服务器、证书和 IAM 角色。有关堆栈创建的资源的完整列表，请参阅软件包自述文件。

如果您是手动设置资源而不是使用 CDK 堆栈，请确保在您的 Amazon MQ 上为 RabbitMQ 代理配置 HTTP 身份验证之前，请确保您有同等的基础架构。

设置 Amazon MQ 的先决条件

AWS CLI 版本 >= 2.28.23，使得在创建代理期间添加用户名和密码成为可选的。

使用 CLI 在 RabbitMQ 中配置 HTTP 身份验证 AWS

此过程使用 AWS CLI 来创建和配置必要的资源。在以下过程中，请确保将占位符值替换为其实际值。

1. 使用 `create-configuration` AWS CLI 命令创建新配置，如以下示例所示。

```
aws mq create-configuration \
  --name "rabbitmq-http-config" \
  --engine-type "RABBITMQ" \
  --engine-version "4.2"
```

此命令返回类似于以下示例的响应。

```
{
    "Arn": "arn:aws:mq:us-west-2:123456789012:configuration:c-fa3390a5-7e01-4559-
ae0c-eb15b38b22ca",
    "AuthenticationStrategy": "simple",
    "Created": "2025-07-17T16:03:01.759943+00:00",
    "Id": "c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca",
    "LatestRevision": {
        "Created": "2025-07-17T16:03:01.759000+00:00",
        "Description": "Auto-generated default for rabbitmq-http-config on RabbitMQ
 4.2",
        "Revision": 1
    },
    "Name": "rabbitmq-http-config"
}
```

2. 创建一个名为的配置文件`rabbitmq.conf`以使用 HTTP 作为身份验证和授权方法，如以下示例所示。将模板中的所有占位符值（标有`${...}`）替换为已部署的 AWS CDK 先决条件堆栈输出或等效基础架构中的实际值。

```
auth_backends.1 = cache
auth_backends.2 = http
auth_cache.cached_backend = http

# HTTP authentication settings
# For more information, see https://github.com/rabbitmq/rabbitmq-auth-backend-http

# FIXME: Replace the ${...} placeholders with actual values
# from your deployed prerequisite CDK stack outputs.
auth_http.http_method = post
auth_http.user_path = ${HttpServerUserPath}
auth_http.vhost_path = ${HttpServerVhostPath}
```

```
auth_http.resource_path = ${HttpServerResourcePath}
auth_http.topic_path = ${HttpServerTopicPath}

# TLS/HTTPS configuration
auth_http.ssl_options.verify = verify_peer
auth_http.ssl_options.sni = test.amazonaws.com

# AWS integration for secure credential retrieval
# For more information, see https://github.com/amazon-mq/rabbitmq-aws

# Replace the ${...} placeholders with actual ARN values
# from your deployed prerequisite CDK stack outputs.
aws.arns.assume_role_arn = ${AmazonMqAssumeRoleArn}
aws.arns.auth_http.ssl_options.cacertfile = ${CaCertArn}
```

3. 使用 update-configuration AWS CLI 命令更新配置。使用步骤 3 中的配置 ID。

```
aws mq update-configuration \
  --configuration-id "<c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca>" \
  --data "$(cat rabbitmq.conf | base64 --wrap=0)"
```

此命令返回类似于以下示例的响应。

```
{
    "Arn": "arn:aws:mq:us-west-2:123456789012:configuration:c-fa3390a5-7e01-4559-
ae0c-eb15b38b22ca",
    "Created": "2025-07-17T16:57:04.520931+00:00",
    "Id": "c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca",
    "LatestRevision": {
        "Created": "2025-07-17T16:57:39.172000+00:00",
        "Revision": 2
    },
    "Name": "rabbitmq-http-config",
    "Warnings": []
}
```

4. 使用 HTTP 配置创建代理。使用前面步骤中的配置 ID 和修订版号。

```
aws mq create-broker \
  --broker-name "rabbitmq-http-test-1" \
  --engine-type "RABBITMQ" \
  --engine-version "4.2" \
  --host-instance-type "mq.m7g.large" \
  --deployment-mode "SINGLE_INSTANCE" \
  --logs '{"General": true}' \
  --publicly-accessible \
  --configuration '{"Id": "<c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca>","Revision":
 <2>}'
```

此命令返回类似于以下示例的响应。

```
{
    "BrokerArn": "arn:aws:mq:us-west-2:123456789012:broker:rabbitmq-http-
test-1:b-2a1b5133-a10c-49d2-879b-8c176c34cf73",
    "BrokerId": "b-2a1b5133-a10c-49d2-879b-8c176c34cf73"
}
```

5. 使用 describe-broker AWS CLI 命令验证代理的状态是否从CREATION_IN_PROGRESS转换为。RUNNING

```
aws mq describe-broker \
  --broker-id "<b-2a1b5133-a10c-49d2-879b-8c176c34cf73>"
```

此命令返回类似于以下示例的响应。config_managed身份验证策略表明代理使用 HTTP 身份验证方法。

```
{
    "AuthenticationStrategy": "config_managed",
    ...,
    "BrokerState": "RUNNING",
    ...
```

```
}
```

6. 使用先决条件 CDK 堆栈创建的测试用户之一验证 RabbitMQ 访问权限

```
# FIXME: Replace ${RabbitMqHttpAuthElbStack.ConsoleUserPasswordArn} with the actual
 ARN from your deployed prerequisite CDK stack outputs
CONSOLE_PASSWORD=$(aws secretsmanager get-secret-value \
  --secret-id ${RabbitMqHttpAuthElbStack.ConsoleUserPasswordArn} \
  --query 'SecretString' --output text)

# FIXME: Replace BrokerConsoleURL with the actual ConsoleURL retrieved by
# calling describe-broker for the broker created above
# Call management API /api/overview (should succeed)
curl -u RabbitMqConsoleUser:$CONSOLE_PASSWORD \
  https://${BrokerConsoleURL}/api/overview

# Try to create a vhost (should fail - console user only has management
 permissions)
curl -u RabbitMqConsoleUser:$CONSOLE_PASSWORD \
  -X PUT https://${BrokerConsoleURL}/api/vhosts/test-vhost \
  -H "Content-Type: application/json" \
  -d '{}'
```

对适用于 RabbitMQ 的亚马逊 MQ 使用 SSL 证书身份验证

本教程介绍如何使用私有证书颁发机构为您的 Amazon MQ for RabbitMQ 代理配置 SSL 证书身份验证。

> **Note**
>
> SSL 证书身份验证插件仅适用于适用于 RabbitMQ 版本 4 及更高版本的亚马逊 MQ。

本页内容

配置 SSL 证书身份验证的先决条件

SSL 证书身份验证使用双向 TLS (mTLS) 对使用 X.509 证书的客户端进行身份验证。您可以通过部署适用于 RabbitMQ mTLS 集成的亚马逊 MQ 的AWS CDK 堆栈来设置本教程中所需的 AWS 资源。

此 CDK 堆栈会自动创建所有必要的 AWS 资源，包括证书颁发机构、客户端证书和 IAM 角色。有关堆栈创建的资源的完整列表，请参阅软件包自述文件。

> **Note**
>
> 在部署 CDK 堆栈之前，请设置RABBITMQ_TEST_USER_NAME环境变量。此值将用作客户端证书中的公用名 (CN)，并且必须与您在教程步骤中使用的用户名相匹配。例如：`export RABBITMQ_TEST_USER_NAME="myuser"`

如果您是手动设置资源而不是使用 CDK 堆栈，请确保在您的 Amazon MQ 上为 RabbitMQ 代理配置 SSL 证书身份验证之前，请确保您有同等的基础架构。

设置 Amazon MQ 的先决条件

AWS CLI 版本 >= 2.28.23，使得在创建代理期间添加用户名和密码成为可选的。

使用 CLI 在 RabbitMQ 中配置 SSL 证书身份验证 AWS

此过程使用 AWS CLI 来创建和配置必要的资源。在以下过程中，请确保将占位符值（例如 ConfigurationID 和 Revision 和）替换为它们的实际值。`<c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca> <2>`

1. 使用 `create-configuration` AWS CLI 命令创建新配置，如以下示例所示。

```
aws mq create-configuration \
  --name "rabbitmq-ssl-config" \
  --engine-type "RABBITMQ" \
  --engine-version "4.2"
```

此命令返回类似于以下示例的响应。

```
{
```

```
    "Arn": "arn:aws:mq:us-west-2:123456789012:configuration:c-fa3390a5-7e01-4559-
ae0c-eb15b38b22ca",
    "AuthenticationStrategy": "simple",
    "Created": "2025-07-17T16:03:01.759943+00:00",
    "Id": "c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca",
    "LatestRevision": {
        "Created": "2025-07-17T16:03:01.759000+00:00",
        "Description": "Auto-generated default for rabbitmq-ssl-config on RabbitMQ
 4.2",
        "Revision": 1
    },
    "Name": "rabbitmq-ssl-config"
}
```

2. 创建名rabbitmq.conf为使用 SSL 证书身份验证的配置文件，如以下示例所示。将模板中的所有占位符值（标有${...}）替换为已部署的 AWS CDK 先决条件堆栈输出或等效基础架构中的实际值。

```
auth_mechanisms.1 = EXTERNAL
ssl_cert_login_from = common_name

auth_backends.1 = internal

# Reject if no client cert
ssl_options.verify = verify_peer
ssl_options.fail_if_no_peer_cert = true

# AWS integration for secure credential retrieval
# For more information, see https://github.com/amazon-mq/rabbitmq-aws

# FIXME: Replace the ${...} placeholders with actual ARN values
# from your deployed prerequisite CDK stack outputs.
aws.arns.assume_role_arn = ${AmazonMqAssumeRoleArn}
aws.arns.ssl_options.cacertfile = ${CaCertArn}
```

3. 使用 update-configuration AWS CLI 命令更新配置，如以下示例所示。在此命令中，添加您在本过程步骤 1 的响应中收到的配置 ID。例如 c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca。

```
aws mq update-configuration \
  --configuration-id "<c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca>" \
  --data "$(cat rabbitmq.conf | base64 --wrap=0)"
```

此命令返回类似于以下示例的响应。

```
{
    "Arn": "arn:aws:mq:us-west-2:123456789012:configuration:c-fa3390a5-7e01-4559-
ae0c-eb15b38b22ca",
    "Created": "2025-07-17T16:57:04.520931+00:00",
    "Id": "c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca",
    "LatestRevision": {
        "Created": "2025-07-17T16:57:39.172000+00:00",
        "Revision": 2
    },
    "Name": "rabbitmq-ssl-config",
    "Warnings": []
}
```

4. 使用您在本过程的步骤 2 中创建的 SSL 证书身份验证配置创建代理。为此，请使用 `create-broker` AWS CLI 命令，如以下示例所示。在此命令中，分别提供您在步骤 1 和步骤 2 的响应中获得的配置 ID 和修订号。例如，`c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca` 和 2。

```
aws mq create-broker \
  --broker-name "rabbitmq-ssl-test-1" \
  --engine-type "RABBITMQ" \
  --engine-version "4.2" \
  --host-instance-type "mq.m7g.large" \
  --deployment-mode "SINGLE_INSTANCE" \
  --logs '{"General": true}' \
  --publicly-accessible \
  --configuration '{"Id": "<c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca>","Revision":
 <2>}' \
  --users '[{"Username":"testuser","Password":"testpassword"}]'
```

此命令返回类似于以下示例的响应。

```
{
    "BrokerArn": "arn:aws:mq:us-west-2:123456789012:broker:rabbitmq-ssl-
test-1:b-2a1b5133-a10c-49d2-879b-8c176c34cf73",
    "BrokerId": "b-2a1b5133-a10c-49d2-879b-8c176c34cf73"
}
```

5. 使用 describe-broker AWS CLI 命令验证代理的状态是否从CREATION_IN_PROGRESS转换为RUNNING，如以下示例所示。在此命令中，提供您在上一步的结果中获得的经纪人 ID。例如 b-2a1b5133-a10c-49d2-879b-8c176c34cf73。

```
aws mq describe-broker \
  --broker-id "<b-2a1b5133-a10c-49d2-879b-8c176c34cf73>"
```

此命令返回类似于以下示例的响应。以下响应是 describe-broker 命令返回的完整输出的缩写版本。此响应显示代理状态以及用于保护代理的认证策略。在这种情况下，config_managed身份验证策略表明代理使用 SSL 证书身份验证方法。

```
{
    "AuthenticationStrategy": "config_managed",
    ...,
    "BrokerState": "RUNNING",
    ...
}
```

6. 使用以下ssl.sh脚本验证 SSL 证书身份验证。

使用此 bash 脚本测试与您的 Amazon MQ for RabbitMQ 代理的连接性。此脚本使用您的客户端证书进行身份验证，并验证连接配置是否正确。如果配置成功，您将看到您的代理发布和使用消息。

如果您收到ACCESS_REFUSED错误，则可以使用代理的 CloudWatch 日志对配置设置进行故障排除。您可以在 Amazon MQ 控制台中找到您的代理的 CloudWatch 日志组链接。

在此脚本中，您需要提供以下值：

- USERNAME：您的客户证书中的常用名 (CN)。
- CLIENT_KEYSTORE：您的客户端密钥库文件的路径（PKCS12 格式）。如果您使用了必备的 CDK 堆栈，则默认路径为$(pwd)/certs/client-keystore.p12。
- KEYSTORE_PASSWORD: 您的客户端密钥库的密码。如果您使用了必备的 CDK 堆栈，则默认密码为changeit。
- BROKER_DNS：您可以在 Amazon MQ 控制台的代理详情页面的 “连接” 下找到此值。

```
#! /bin/bash
set -e

# Client information
## FIXME: Update this value with the client ID and secret of your confidential
 application client
USERNAME=<client_cert_common_name>
CLIENT_KEYSTORE=$(pwd)/certs/client-keystore.p12
KEYSTORE_PASSWORD=changeit

BROKER_DNS=<broker_dns>
CONNECTION_STRING=amqps://${BROKER_DNS}:5671

# Produce/consume messages using the above connection string
QUEUES_COUNT=1
PRODUCERS_COUNT=1
CONSUMERS_COUNT=1
PRODUCER_RATE=1

finch run --rm --ulimit nofile=40960:40960 \
    -v ${CLIENT_KEYSTORE}:/certs/client-keystore.p12:ro \
    -e JAVA_TOOL_OPTIONS="-Djavax.net.ssl.keyStore=/certs/client-
keystore.p12 -Djavax.net.ssl.keyStorePassword=${KEYSTORE_PASSWORD} -
Djavax.net.ssl.keyStoreType=PKCS12" \
    pivotalrabbitmq/perf-test:latest \
    --queue-pattern 'test-queue-cert-%d' --queue-pattern-from 1 --queue-pattern-to
 $QUEUES_COUNT \
    --producers $PRODUCERS_COUNT --consumers $CONSUMERS_COUNT \
    --id "cert-test${QUEUES_COUNT}q${PRODUCERS_COUNT}p${CONSUMERS_COUNT}c
${PRODUCER_RATE}r" \
    --uri ${CONNECTION_STRING} \
    --sasl-external \
    --use-default-ssl-context \
```

```
    --flag persistent --rate $PRODUCER_RATE
```

将 mTLS 用于 AMQP 和管理端点

本教程介绍如何使用私有证书颁发机构为 AMQP 客户端连接和 RabbitMQ 管理接口配置双向 TLS (mTLS)。

> **Note**
>
> 只有适用于 RabbitMQ 版本 4 及更高版本的 Amazon MQ 才能使用私有证书颁发机构。

本页内容

配置 mTLS 的先决条件

您可以通过部署适用于 RabbitMQ 的 Amazon MQ 的AWS CDK 堆栈来设置本教程中所需的 AWS 资源。

此 CDK 堆栈会自动创建所有必要的 AWS 资源，包括证书颁发机构、客户端证书和 IAM 角色。有关堆栈创建的资源的完整列表，请参阅软件包自述文件。

如果您是手动设置资源而不是使用 CDK 堆栈，请确保在您的 Amazon MQ 上为 RabbitMQ 经纪商配置 mTLs 之前，请确保您有同等的基础架构。

设置 Amazon MQ 的先决条件

AWS CLI 版本 >= 2.28.23，使得在创建代理期间添加用户名和密码成为可选的。

使用 CLI 在 RabbitMQ 中配置 mTLS AWS

此过程使用 AWS CLI 来创建和配置必要的资源。在以下过程中，请确保将占位符值（例如 ConfigurationID 和 Revision 和）替换为它们的实际值。`<c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca>` `<2>`

1. 使用 `create-configuration` AWS CLI 命令创建新配置，如以下示例所示。

```
aws mq create-configuration \
  --name "rabbitmq-mtls-config" \
  --engine-type "RABBITMQ" \
  --engine-version "4.2"
```

此命令返回类似于以下示例的响应。

```
{
    "Arn": "arn:aws:mq:us-west-2:123456789012:configuration:c-fa3390a5-7e01-4559-
ae0c-eb15b38b22ca",
    "AuthenticationStrategy": "simple",
    "Created": "2025-07-17T16:03:01.759943+00:00",
    "Id": "c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca",
    "LatestRevision": {
        "Created": "2025-07-17T16:03:01.759000+00:00",
        "Description": "Auto-generated default for rabbitmq-mtls-config on RabbitMQ
 4.2",
        "Revision": 1
    },
    "Name": "rabbitmq-mtls-config"
}
```

2. 创建一个名为的配置文件`rabbitmq.conf`，用于为 AMQP 和管理端点配置 mTLS，如以下示例所示。将模板中的所有占位符值（标有`${...}`）替换为已部署的 AWS CDK 先决条件堆栈输出或等效基础架构中的实际值。

```
auth_backends.1 = internal

# TLS configuration
ssl_options.verify = verify_peer
ssl_options.fail_if_no_peer_cert = true
management.ssl.verify = verify_peer

# AWS integration for secure credential retrieval
# For more information, see https://github.com/amazon-mq/rabbitmq-aws
```

```
# FIXME: Replace the ${...} placeholders with actual ARN values
# from your deployed prerequisite CDK stack outputs.
aws.arns.assume_role_arn = ${AmazonMqAssumeRoleArn}
aws.arns.ssl_options.cacertfile = ${CaCertArn}
aws.arns.management.ssl.cacertfile = ${CaCertArn}
```

3. 使用 update-configuration AWS CLI 命令更新配置，如以下示例所示。在此命令中，添加您在本过程步骤 1 的响应中收到的配置 ID。例如 c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca。

```
aws mq update-configuration \
  --configuration-id "<c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca>" \
  --data "$(cat rabbitmq.conf | base64 --wrap=0)"
```

此命令返回类似于以下示例的响应。

```
{
    "Arn": "arn:aws:mq:us-west-2:123456789012:configuration:c-fa3390a5-7e01-4559-
ae0c-eb15b38b22ca",
    "Created": "2025-07-17T16:57:04.520931+00:00",
    "Id": "c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca",
    "LatestRevision": {
        "Created": "2025-07-17T16:57:39.172000+00:00",
        "Revision": 2
    },
    "Name": "rabbitmq-mtls-config",
    "Warnings": []
}
```

4. 使用您在本过程的步骤 2 中创建的 mTLS 配置创建经纪商。为此，请使用 create-broker AWS CLI 命令，如以下示例所示。在此命令中，分别提供您在步骤 1 和步骤 2 的响应中获得的配置 ID 和修订号。例如，c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca 和 2。

```
aws mq create-broker \
  --broker-name "rabbitmq-mtls-test-1" \
  --engine-type "RABBITMQ" \
```

```
  --engine-version "4.2" \
  --host-instance-type "mq.m7g.large" \
  --deployment-mode "SINGLE_INSTANCE" \
  --logs '{"General": true}' \
  --publicly-accessible \
  --configuration '{"Id": "<c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca>","Revision":
 <2>}' \
  --users '[{"Username":"testuser","Password":"testpassword"}]'
```

此命令返回类似于以下示例的响应。

```
{
    "BrokerArn": "arn:aws:mq:us-west-2:123456789012:broker:rabbitmq-mtls-
test-1:b-2a1b5133-a10c-49d2-879b-8c176c34cf73",
    "BrokerId": "b-2a1b5133-a10c-49d2-879b-8c176c34cf73"
}
```

5. 使用 describe-broker AWS CLI 命令验证代理的状态是否从CREATION_IN_PROGRESS转换为RUNNING，如以下示例所示。在此命令中，提供您在上一步的结果中获得的经纪人 ID。例如 b-2a1b5133-a10c-49d2-879b-8c176c34cf73。

```
aws mq describe-broker \
  --broker-id "<b-2a1b5133-a10c-49d2-879b-8c176c34cf73>"
```

此命令返回类似于以下示例的响应。以下响应是 describe-broker 命令返回的完整输出的缩写版本。

```
{
    "AuthenticationStrategy": "simple",
    ...,
    "BrokerState": "RUNNING",
    ...
}
```

6. 使用以下mtls.sh脚本验证 mTLS 身份验证。

使用此 bash 脚本测试与您的 Amazon MQ for RabbitMQ 代理的连接性。此脚本使用您的客户端证书进行身份验证并验证连接配置是否正确。如果配置成功，您将看到您的代理发布和使用消息。

如果您收到ACCESS_REFUSED错误，则可以使用代理的 CloudWatch 日志对配置设置进行故障排除。您可以在 Amazon MQ 控制台中找到您的代理的 CloudWatch 日志组链接。

在此脚本中，您需要提供以下值：

- USERNAME和PASSWORD：您使用代理创建的 RabbitMQ 用户证书。
- CLIENT_KEYSTORE：您的客户端密钥库文件的路径（PKCS12 格式）。如果您使用了必备的 CDK 堆栈，则默认路径为$(pwd)/certs/client-keystore.p12。
- KEYSTORE_PASSWORD: 您的客户端密钥库的密码。如果您使用了必备的 CDK 堆栈，则默认密码为changeit。
- BROKER_DNS：您可以在 Amazon MQ 控制台的代理详情页面的 “连接” 下找到此值。

```
#! /bin/bash
set -e

# Client information
## FIXME: Update this value with the client ID and secret of your confidential
 application client
USERNAME=<testuser>
PASSWORD=<testpassword>
CLIENT_KEYSTORE=$(pwd)/certs/client-keystore.p12
KEYSTORE_PASSWORD=changeit

BROKER_DNS=<broker_dns>
CONNECTION_STRING=amqps://${USERNAME}:${PASSWORD}@${BROKER_DNS}:5671

# Produce/consume messages using the above connection string
QUEUES_COUNT=1
PRODUCERS_COUNT=1
CONSUMERS_COUNT=1
PRODUCER_RATE=1

finch run --rm --ulimit nofile=40960:40960 \
    -v ${CLIENT_KEYSTORE}:/certs/client-keystore.p12:ro \
```

```
    -e JAVA_TOOL_OPTIONS="-Djavax.net.ssl.keyStore=/certs/client-
keystore.p12 -Djavax.net.ssl.keyStorePassword=${KEYSTORE_PASSWORD} -
Djavax.net.ssl.keyStoreType=PKCS12" \
    pivotalrabbitmq/perf-test:latest \
    --queue-pattern 'test-queue-cert-%d' --queue-pattern-from 1 --queue-pattern-to
 $QUEUES_COUNT \
    --producers $PRODUCERS_COUNT --consumers $CONSUMERS_COUNT \
    --id "cert-test${QUEUES_COUNT}q${PRODUCERS_COUNT}p${CONSUMERS_COUNT}c
${PRODUCER_RATE}r" \
    --uri ${CONNECTION_STRING} \
    --use-default-ssl-context \
    --flag persistent --rate $PRODUCER_RATE
```

连接您的 JMS 应用程序

本教程向您展示如何使用 RabbitMQ JMS 客户端将 JMS 应用程序连接到 Amazon MQ for RabbitMQ 代理。您将学习如何创建用于发送消息的创建器和用于接收来自 RabbitMQ 队列的消息的使用者。

在开始之前，请将相应的 RabbitMQ JMS 依赖项添加到你的 Maven 项目中：

对于 JMS 1.1 和 2.0：

```
<dependencies>

  <dependency>
    <groupId>com.rabbitmq.jms</groupId>
    <artifactId>rabbitmq-jms</artifactId>
    <version>2.12.0</version>
  </dependency>

</dependencies>
```

对于 JMS 3.1：

```
<dependencies>

  <dependency>
    <groupId>com.rabbitmq.jms</groupId>
    <artifactId>rabbitmq-jms</artifactId>
    <version>3.5.0</version>
  </dependency>
```

```
</dependencies>
```

创建制作人

以下代码示例显示了如何使用 JMS 写入 RabbitMQ 队列：

```
import jakarta.jms.*;
import com.rabbitmq.jms.admin.*;

// Setting the connection factory
RMQConnectionFactory factory = new RMQConnectionFactory();
factory.setHost(envProps.getProperty("RABBITMQ_HOST", "localhost"));
factory.setPort(Integer.parseInt(envProps.getProperty("RABBITMQ_PORT", "5672")));
factory.setUsername(envProps.getProperty("RABBITMQ_USERNAME", "guest"));
factory.setPassword(envProps.getProperty("RABBITMQ_PASSWORD", "guest"));
factory.setVirtualHost(envProps.getProperty("RABBITMQ_VIRTUAL_HOST", "/"));
factory.useSslProtocol();

connection = factory.createConnection();
connection.start();

String queueName = "test-queue-jms";
Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);

RMQDestination destination = new RMQDestination(queueName, true, false);

// Send the message to the queue
MessageProducer producer = session.createProducer(destination);
producer.setDeliveryMode(DeliveryMode.PERSISTENT);

String msg_content = "Hello World!!";
TextMessage textMessage = session.createTextMessage(msg_content);
producer.send(textMessage);

System.out.printf("Published to AMQP queue '%s': %s", queueName, msg_content);
```

创建消费者

以下代码示例显示了如何使用 JMS 从 RabbitMQ 队列中读取：

```
import jakarta.jms.*;
import com.rabbitmq.jms.admin.*;
```

```
// Setting the connection factory
RMQConnectionFactory factory = new RMQConnectionFactory();
factory.setHost(envProps.getProperty("RABBITMQ_HOST", "localhost"));
factory.setPort(Integer.parseInt(envProps.getProperty("RABBITMQ_PORT", "5672")));
factory.setUsername(envProps.getProperty("RABBITMQ_USERNAME", "guest"));
factory.setPassword(envProps.getProperty("RABBITMQ_PASSWORD", "guest"));
factory.setVirtualHost(envProps.getProperty("RABBITMQ_VIRTUAL_HOST", "/"));
factory.useSslProtocol();

// Establish the connection and session
jakarta.jms.Connection connection = factory.createConnection();

String queueName = "test-queue-jms";
Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);

RMQDestination destination = new RMQDestination();
destination.setDestinationName(queueName);
destination.setAmqp(true);
destination.setAmqpQueueName(queueName);

// Initialize consumer
MessageConsumer consumer = session.createConsumer(destination);
consumer.setMessageListener(message -> {
    try {
        if (message instanceof TextMessage) {
            TextMessage textMessage = (TextMessage) message;
            System.out.printf("Message: %s%n", textMessage.getText());
        } else if (message instanceof BytesMessage) {
            BytesMessage bytesMessage = (BytesMessage) message;
            byte[] bytes = new byte[(int) bytesMessage.getBodyLength()];
            bytesMessage.readBytes(bytes);
            String content = new String(bytes);
            System.out.printf("Message: %s%n", content);
        } else {
            System.out.printf("Message: [%s]%n", message.getClass().getSimpleName());
        }
    } catch (JMSException e) {
        System.err.printf("Error processing message: %s%n", e.getMessage());
    }
});

connection.start();
```

Amazon MQ 中的安全性

云安全 AWS 是重中之重。作为 AWS 客户，您可以受益于专为满足大多数安全敏感型组织的要求而构建的数据中心和网络架构。

安全是双方共同承担 AWS 的责任。责任共担模式将其描述为云的安全性和云中的安全性：

- 云安全 — AWS 负责保护在 AWS 云中运行 AWS 服务的基础架构。 AWS 还为您提供可以安全使用的服务。作为AWS 合规计划合规计划合规计划合的一部分，第三方审计师定期测试和验证我们安全的有效性。要了解适用于 Amazon MQ 的合规计划，请参阅按合规计划提供的范围内的AWS 服务按合规计划划。
- 云端安全-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。您还将学习如何使用其他 AWS 服务来帮助您监控和保护您的 Amazon MQ 资源。

主题

Amazon MQ 中的数据保护

责任 AWS 共担模式适用于 Amazon MQ 中的数据保护。如本模型所述 AWS ，负责保护运行所有内容的全球基础架构 AWS Cloud。您负责维护对托管在此基础结构上的内容的控制。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解答AWS条款。 有关欧洲数据保护的信息，请参阅通用数据保护条例（GDPR）中心。

出于数据保护目的，我们建议您保护 AWS 账户 凭证并使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 设置个人用户。这样，每个用户只获得履行其工作职责所需的权限。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）。
- 用于 SSL/TLS 与 AWS 资源通信。我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 使用设置 API 和用户活动日志 AWS CloudTrail。有关使用 CloudTrail 跟踪捕获 AWS 活动的信息，请参阅《AWS CloudTrail 用户指南》中的使用跟 CloudTrail 踪。
- 使用 AWS 加密解决方案以及其中的所有默认安全控件 AWS 服务。
- 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。
- 如果您在 AWS 通过命令行界面或 API 进行访问时需要经过 FIPS 140-3 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 140-3 版》https://aws.amazon.com/compliance/fips/。

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（如名称字段）。这包括您使用控制台、API 或软件开发工具包 AWS 服务 使用 Amazon MQ 或其他软件开发工具包 AWS CLI的情况。 AWS 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志。如果您向外部服务器提供 URL，强烈建议您不要在网址中包含凭证信息来验证对该服务器的请求。

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。代理名称和用户名可供其他 AWS 服务（包括日 CloudWatch 志）访问。代理用户名不适合用于私有或敏感数据。

> **Important**
>
> TLS 1.3 不适用于 RabbitMQ 代理。

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，请参见AWS Encryption SDK 开发人员指南。

静态加密

Amazon MQ 与 AWS Key Management Service (KMS) 集成，提供透明的服务器端加密。Amazon MQ 始终加密您的静态数据。

当你为 ActiveMQ 代理创建亚马逊 MQ 或为 RabbitMQ 代理创建亚马逊 MQ 时，你可以指定你希望 Amazon MQ 用来 AWS KMS key 加密静态数据。如果您未指定 KMS 密钥，Amazon MQ 会为您创建自己的 AWS KMS 密钥并代表您使用该密钥。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 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。

> **Important**
>
> - 撤消授权的操作无法撤消。删除代理以撤销访问权限。
> - 对于使用 Amazon Elastic File System (EFS) 存储消息数据的 Amazon MQ for ActiveMQ 代理，在采取必要操作后，撤销使用您账户中 KMS 密钥的权限可能需要几个小时。
> - 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。
> - 在采取必要操作后，停用密钥或撤销授权可能需要几个小时。
> - 对于加密或解密 CloudWatch 日志，您无法配置 Amazon MQ 使用的加密密钥。CloudWatch 日志使用加密保护静态数据，日志组是加密的。默认情况下， CloudWatch 日

志服务管理服务器端加密。有关如何加密日志组的更多信息，请参阅 Amazon CloudWatch 日志用户指南。

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

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 集群复制流量仅通过您的代理的 VPC，并且 AWS 数据中心之间的所有网络流量都在物理层进行透明加密。适用于 RabbitMQ 的 Amazon MQ 集群代理目前不支持对集群复制进行加密。Inter-node 要了解有关传输中数据的更多信息，请参阅加密 Data-at-Rest 和传输中。

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
- MQTT 结束了 WebSocket
- OpenWire
- STOMP
- 大吃一惊 WebSocket

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) AWS 服务 可帮助管理员安全地控制对 AWS 资源的访问权限。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。您可以使用 IAM AWS 服务 ，无需支付额外费用。

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
- 经纪人身份验证和授权
- 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

您的使用方式 AWS Identity and Access Management (IAM) 因您的角色而异：

- 服务用户：如果您无法访问功能，请从管理员处请求权限（请参阅Amazon MQ 身份和访问问题排查）
- 服务管理员：确定用户访问权限并提交权限请求（请参阅Amazon MQ 如何与 IAM 协同工作）
- IAM 管理员：编写用于管理访问权限的策略（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 AWS 使用身份凭证登录的方式。您必须以 IAM 用户身份进行身份验证 AWS 账户根用户，或者通过担任 IAM 角色进行身份验证。

您可以使用来自身份源的证书 AWS IAM Identity Center（例如（IAM Identity Center）、单点登录身份验证或 Google/Facebook 证书，以联合身份登录。有关登录的更多信息，请参阅《AWS 登录 用户指南》中的如何登录您的 AWS 账户。

对于编程访问，AWS 提供 SDK 和 CLI 来对请求进行加密签名。有关更多信息，请参阅《IAM 用户指南》中的适用于 API 请求的AWS 签名版本 4。

AWS 账户 根用户

创建时 AWS 账户，首先会有一个名为 AWS 账户 root 用户的登录身份，该身份可以完全访问所有资源 AWS 服务 和资源。我们强烈建议不要使用根用户进行日常任务。有关要求根用户凭证的任务，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 用户是对某个人员或应用程序具有特定权限的一个身份。建议使用临时凭证，而非具有长期凭证的 IAM 用户。有关更多信息，请参阅 IAM 用户指南中的要求人类用户使用身份提供商的联合身份验证才能 AWS 使用临时证书进行访问。

IAM 组指定一组 IAM 用户，便于更轻松地对大量用户进行权限管理。有关更多信息，请参阅《IAM 用户指南》中的 IAM 用户使用案例。

IAM 角色

IAM 角色是具有特定权限的身份，可提供临时凭证。您可以通过从用户切换到 IAM 角色（控制台）或调用 AWS CLI 或 AWS API 操作来代入角色。有关更多信息，请参阅《IAM 用户指南》中的担任角色的方法。

IAM 角色对于联合用户访问、临时 IAM 用户权限、跨账户访问、跨服务访问以及在 Amazon EC2 上运行的应用程序非常有用。有关更多信息，请参阅《IAM 用户指南》中的 IAM 中的跨账户资源访问。

使用策略管理访问

您可以 AWS 通过创建策略并将其附加到 AWS 身份或资源来控制中的访问权限。策略定义了与身份或资源关联时的权限。 AWS 在委托人提出请求时评估这些政策。大多数策略都以 JSON 文档的 AWS 形式存储在中。有关 JSON 策略文档的更多信息，请参阅《IAM 用户指南》中的 JSON 策略概述。

管理员使用策略，通过定义哪个主体可以在什么条件下对哪些资源执行哪些操作来指定谁有权访问什么。

默认情况下，用户和角色没有权限。IAM 管理员创建 IAM 策略并将其添加到角色中，然后用户可以担任这些角色。IAM 策略定义权限，与执行操作所用的方法无关。

Identity-based 政策

Identity-based 策略是您附加到身份（用户、组或角色）的 JSON 权限策略文档。这些策略控制身份可以执行什么操作、对哪些资源执行以及在什么条件下执行。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的使用客户管理型策略定义自定义 IAM 权限。

Identity-based 策略可以是内联策略（直接嵌入到单个身份中）或托管策略（附加到多个身份的独立策略）。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。

Resource-based 政策

Resource-based 策略是您附加到资源的 JSON 策略文档。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。您必须在基于资源的策略中指定主体。

Resource-based 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 IAM 中的 AWS 托管策略。

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，但它们不使用 JSON 策略文档格式。

Amazon S3 和 Amazon VPC 就是支持 ACL 的服务示例。 AWS WAF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 支持其他策略类型，这些策略类型可以设置更常见的策略类型授予的最大权限：

- 权限边界 – 设置基于身份的策略可以授予 IAM 实体的最大权限。有关更多信息，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）– 指定 AWS Organizations中组织或组织单元的最大权限。有关更多信息，请参阅《AWS Organizations 用户指南》中的服务控制策略。
- 资源控制策略（RCP）– 设置对账户中资源的最大可用权限。有关更多信息，请参阅《AWS Organizations 用户指南》中的资源控制策略（RCP）。
- 会话策略 – 在为角色或联合用户创建临时会话时，作为参数传递的高级策略。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。要了解在涉及多种策略类型时如何 AWS 确定是否允许请求，请参阅 IAM 用户指南中的策略评估逻辑。

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。要全面了解 Amazon MQ 和其他 AWS 服务如何与 IAM 配合使用，请参阅 IAM 用户指南中的与 IAM 配合使用的AWS 服务。

亚马逊 MQ 使用 IAM for Amazon MQ API 操作来创建、更新、删除和列出经纪人。要让代理访问发布和订阅消息，Amazon MQ for ActiveMQ 支持原生 ActiveMQ 身份验证和 LDAP，而适用于 RabbitMQ 的亚马逊 MQ 支持 IAM 身份验证和其他方法。有关更多信息，请参阅 the section called “经纪人身份验证和授权”。

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。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。在策略中包含操作以授予执行关联操作的权限。

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。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。作为最佳实践，请使用其 Amazon 资源名称（ARN）指定资源。对于不支持资源级权限的操作，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。Amazon MQ 代理和配置都有与之关联的唯一亚马逊资源名称 (ARNs)，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}

有关格式的更多信息 ARNs，请参阅 Amazon 资源名称 (ARNs) 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 ARNs 用逗号分隔。

```
"Resource": [
      "resource1",
      "resource2"
```

要查看 Amazon MQ 资源类型及其列表 ARNs，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。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。

`Condition` 元素根据定义的条件指定语句何时执行。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。要查看所有 AWS 全局条件键，请参阅 IAM 用户指南中的AWS 全局条件上下文密钥。

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	Type
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
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：

JSON

```
{
    "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 Amazon MQ 资源添加标签
- 使用 IAM 标签控制访问

Amazon MQ IAM 角色

I AM 角色是您的 AWS 账户中具有特定权限的实体。

将临时凭证用于 Amazon MQ

可以使用临时凭证进行联合身份验证登录，分派 IAM 角色或分派跨账户角色。您可以通过调用AssumeRole或之类的 AWS STS API 操作来获取临时安全证书GetFederationToken。

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他服务中的资源以代表您完成操作。服务角色显示在 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。他们也无法使用 AWS 管理控制台 AWS CLI、或 AWS API 执行任务。IAM 管理员必须创建 IAM 策略，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，请遵循以下指南和建议：

- 开始使用 AWS 托管策略并转向最低权限权限 — 要开始向用户和工作负载授予权限，请使用为许多常见用例授予权限的AWS 托管策略。它们在你的版本中可用 AWS 账户。我们建议您通过定义针对您的用例的 AWS 客户托管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 策略设置权限时，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 策略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。如果服务操作是通过特定 AWS 服务的（例如）使用的，则也可以使用条件来授予对服务操作的访问权限 CloudFormation。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 策略，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 策略语言（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》中的使用 IAM Access Analyzer 验证策略。
- 需要多重身份验证 (MFA)-如果 AWS 账户您的场景需要 IAM 用户或根用户，请启用 MFA 以提高安全性。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》中的使用 MFA 保护 API 访问。

有关 IAM 中的最佳实操的更多信息，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。

为确保这些实体仍然可以使用 Amazon MQ 控制台，还要将以下 AWS 托管策略附加到这些实体。有关更多信息，请参阅《IAM 用户指南》中的为用户添加权限：

```
AmazonMQReadOnlyAccess
```

对于仅调用 AWS CLI 或 AWS API 的用户，您无需为其设置最低控制台权限。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或使用 AWS CLI 或 AWS API 以编程方式完成此操作的权限。

```
{
    "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

亚马逊 MQ 使用标准 AWS 请求签名进行 API 身份验证。有关更多信息，请参阅 AWS 一般参考 中的签署 AWS 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，您必须编辑您的 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 附加权限参考
- Resource-level 亚马逊 MQ API 操作的权限

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。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Action": [
            "mq:*",
            "ec2:CreateNetworkInterface",
            "ec2:DeleteNetworkInterface",
            "ec2:DetachNetworkInterface",
            "ec2:DescribeInternetGateways",
            "ec2:DescribeNetworkInterfaces",
            "ec2:DescribeRouteTables",
            "ec2:DescribeSecurityGroups",
            "ec2:DescribeSubnets",
            "ec2:DescribeVpcs"
        ],
        "Effect": "Allow",
        "Resource": "*"
    },{
        "Action": [
            "ec2:CreateNetworkInterfacePermission",
            "ec2:DeleteNetworkInterfacePermission",
            "ec2:DescribeNetworkInterfacePermissions"
        ],
        "Effect": "Allow",
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "ec2:AuthorizedService": "mq.amazonaws.com"
            }
        }
    }]
}
```

有关更多信息，请参阅Setting Up Amazon MQ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
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 附加权限参考

下表列出了 Amazon MQ API 及特定功能（如 OAuth 2.0 认证）所需的附加 IAM 权限。

Amazon MQ REST API	权限	说明
UpdateBroker	`mq:UpdateBrokerAccessConfiguration`	需要此权限以更新关联代理配置中的认证和授权选项。有关更多信息，请参阅 OAuth 适用于 RabbitMQ 的亚马逊 MQ 的 2.0 身份验证和授权。

Resource-level 亚马逊 MQ API 操作的权限

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*

API 操作	资源类型（* 为必需）
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*
RebootBroker	代理*
UpdateBroker	代理*
UpdateConfiguration	配置*
UpdateUser	代理*

经纪人身份验证和授权

Amazon MQ 根据您的经纪商引擎类型提供不同的身份验证和授权方法。

适用于 ActiveMQ 的亚马逊 MQ 的身份验证和授权

适用于 ActiveMQ 的亚马逊 MQ 支持以下身份验证和授权方法：

简单认证与授权

在这种方法中，代理用户是通过 Amazon MQ 控制台或 API 创建和管理的。可以为用户配置访问队列、主题和 ActiveMQ Web 控制台的特定权限。有关此方法的更多信息，请参阅创建 ActiveMQ 代理用户。

LDAP 身份验证和授权

在这种方法中，代理用户通过存储在您的 LDAP 服务器中的凭据进行身份验证。您可以通过 LDAP 服务器添加、删除和修改用户以及为主题和队列分配权限，从而提供集中式身份验证和授权。有关此方法的更多信息，请参阅将 ActiveMQ 代理与 LDAP 集成。

适用于 RabbitMQ 的亚马逊 MQ 的身份验证和授权

Amazon MQ for RabbitMQ 支持以下认证与授权方法：

简单认证与授权

在此方法中，代理用户存储在 RabbitMQ 代理内部，并通过 Web 控制台或管理 API 进行管理。虚拟主机、交换机、队列和主题的权限直接在 RabbitMQ 中配置。这是默认方法。有关更多信息，请参阅简单身份验证和授权。

OAuth 2.0 认证与授权

在此方法中，代理用户及其权限由外部 OAuth 2.0 身份提供者（IdP）管理。虚拟主机、交换机、队列和主题的用户认证和资源权限通过 OAuth 2.0 提供程序的范围系统进行集中管理。这简化了用户管理，并实现了与现有身份系统的集成。有关更多信息，请参阅 OAuth 2.0 身份验证和授权。

IAM 身份验证和授权

在此方法中，代理用户通过 IAM 出站联合使用 AWS IAM 凭证进行身份验证。IAM 证书用于从 AWS 安全令牌服务 (STS) 获取 JWT 令牌，而这些 JWT 令牌则用作 OAuth 2.0 令牌进行身份验证。此方法利用了亚马逊 MQ 中对 RabbitMQ 的现有 OAuth 2.0 支持，RabbitMQ 充当 OAuth 2.0 身份提供商。 AWS 用户身份验证由 AWS IAM 处理，而虚拟主机、交易所、队列和主题的资源权限则通过 RabbitMQ 中配置的 IAM 策略和范围别名进行管理。有关更多信息，请参阅 IAM 身份验证和授权。

LDAP 身份验证和授权

在这种方法中，代理用户及其权限由外部 LDAP 目录服务管理。用户身份验证和资源权限通过 LDAP 服务器集中管理，允许用户使用其现有的目录服务凭据访问 RabbitMQ。有关更多信息，请参阅 LDAP 身份验证和授权。

HTTP 身份验证和授权

在这种方法中，代理用户及其权限由外部 HTTP 服务器管理。用户身份验证和资源权限通过 HTTP 服务器集中管理，允许用户使用自己的身份验证和授权提供程序访问 RabbitMQ。有关此方法的更多信息，请参阅 HTTP 身份验证和授权。

SSL 证书身份验证

亚马逊 MQ 支持 RabbitMQ 经纪商的双向 TLS (mTLS)。SSL 身份验证插件使用来自 mTLS 连接的客户端证书对用户进行身份验证。在这种方法中，使用 X.509 客户端证书而不是用户名和密码凭证对经纪人用户进行身份验证。根据受信任的证书颁发机构 (CA) 对客户端的证书进行验证，用户名是从证书的字段（例如公用名 (CN) 或主题备用名称 (SAN)）中提取的。此方法无需通过网络传输凭据即可提供强身份验证。有关更多信息，请参阅 SSL 证书身份验证。

> **Note**
>
> RabbitMQ 支持同时使用多种身份验证和授权方法。例如，您可以同时启用 OAuth 2.0 和简单（内部）身份验证。有关更多信息，请参阅 OAuth 2.0 教程中关于同时启用 OAuth 2.0 和简单（内部）身份验证的部分以及 Rabb itMQ 访问控制文档。

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使用案例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。

Amazon MQ 支持以下 AWS 托管策略：

- AmazonMQApiFullAccess
- AmazonMQApiReadOnlyAccess

- 亚马逊MQFull访问权限
- AmazonMQReadOnlyAccess
- AmazonMQServiceRolePolicy

AWS 托管策略：Amazon MQService RolePolicy

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

亚马逊 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。服务相关角色由 Amazon MQ 预定义，包括该服务代表您调用 AWS 其他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他 IAM 实体的权限策略。

只有在首先删除相关资源后，您才能删除服务关联角色。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，并查找 Service-Linked Role（服务相关角色）列中显示为 Yes（是）的服务。选择是和链接，查看该服务的服务关联角色文档。

Amazon MQ 的服务相关角色权限

Amazon MQ 使用名为 MQ 的服务相关角色 AWSServiceRoleForAmazon—— 亚马逊 MQ 使用此服务相关角色代表您调用服务。 AWS

AWSServiceRoleForAmazonMQ 服务相关角色信任以下服务来代入该角色：

- `mq.amazonaws.com`

Amazon MQ 使用附加到 AWSService RoleForAmazon MQ 服务相关角色的权限策略AmazonMQServiceRolePolicy对指定资源完成以下操作：

- 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 日志。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeVpcEndpoints"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateVpcEndpoint"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:vpc/*",
                "arn:aws:ec2:*:*:subnet/*",
                "arn:aws:ec2:*:*:security-group/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateVpcEndpoint"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:vpc-endpoint/*"
            ],
            "Condition": {
```

```
                "StringEquals": {
                    "aws:RequestTag/AMQManaged": "true"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": "arn:aws:ec2:*:*:vpc-endpoint/*",
            "Condition": {
                "StringEquals": {
                    "ec2:CreateAction": "CreateVpcEndpoint"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DeleteVpcEndpoints"
            ],
            "Resource": "arn:aws:ec2:*:*:vpc-endpoint/*",
            "Condition": {
                "StringEquals": {
                    "ec2:ResourceTag/AMQManaged": "true"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents",
                "logs:DescribeLogStreams",
                "logs:DescribeLogGroups",
                "logs:CreateLogStream",
                "logs:CreateLogGroup"
            ],
            "Resource": [
                "arn:aws:logs:*:*:log-group:/aws/amazonmq/*"
            ]
        }
    ]
```

```
}
```

您必须配置权限，允许 IAM 实体（如用户、组或角色）创建、编辑或删除服务关联角色。有关更多信息，请参阅《IAM 用户指南》中的服务关联角色权限。

为 Amazon MQ 创建服务相关角色

您无需手动创建服务关联角色。当您首次创建代理时，Amazon MQ 会创建一个服务相关角色来代表您调用 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，此服务关联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 账户中的新角色。

如果您删除该服务关联角色，然后需要再次创建，您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，使用服务名称创建服务相关角色。`mq.amazonaws.com`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，可以使用同样的过程再次创建角色。

> **Important**
>
> 服务关联角色仅针对 Amazon MQ for RabbitMQ 创建。

为 Amazon MQ 编辑服务相关角色

Amazon MQ 不允许您编辑 AWSService RoleForAmazon MQ 服务相关角色。不过，您可以使用 IAM 编辑角色的说明。有关更多信息，请参阅《IAM 用户指南》中的编辑服务关联角色。

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关联角色的功能或服务，我们建议您删除该角色。这样就没有未被主动监控或维护的未使用实体。但是，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 MQ 使用的亚马逊 MQ 资源 AWSService RoleForAmazon

- 使用、亚马逊 MQ CLI 或亚马逊 MQ AWS 管理控制台 API 删除您的亚马逊 MQ 经纪商。有关删除代理的更多信息，请参阅???。

使用 IAM 手动删除服务关联角色

使用 IAM 控制台 AWS CLI、或 AWS API 删除 AWSService RoleForAmazon MQ 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 区域和端点。

区域名称	区域标识	Amazon MQ 支持
美国东部（弗吉尼亚州北部）	us-east-1	是
美国东部（俄亥俄州）	us-east-2	是
美国西部（北加利福尼亚）	us-west-1	是
美国西部（俄勒冈州）	us-west-2	是
亚太地区（孟买）	ap-south-1	是
亚太地区（大阪）	ap-northeast-3	是
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

区域名称	区域标识	Amazon MQ 支持
亚太地区（东京）	ap-northeast-1	是
加拿大（中部）	ca-central-1	是
欧洲地区（法兰克福）	eu-central-1	是
欧洲地区（爱尔兰）	eu-west-1	是
欧洲地区（伦敦）	eu-west-2	是
欧洲地区（巴黎）	eu-west-3	是
南美洲（圣保罗）	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
- 我想允许 AWS 账户之外的用户访问我的 Amazon MQ 资源

我无权在 Amazon MQ 中执行操作

如果 AWS 管理控制台 告诉您您无权执行某项操作，则必须联系管理员寻求帮助。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某的详细信息*widget*但没有mq:*GetWidget*权限时，就会出现以下示例错误。

```
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给该服务，而不是创建新的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。您的管理员是提供登录凭证的人。

我想允许 AWS 账户之外的用户访问我的 Amazon MQ 资源

您可以创建一个角色，以便其他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，可以代入角色。对于支持基于资源的策略或访问控制列表 (ACLs) 的服务，您可以使用这些策略向人们授予访问您的资源的权限。

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。
- 要了解如何提供对您拥有的资源的访问权限 AWS 账户 ，请参阅 IAM 用户指南中的向您拥有 AWS 账户 的另一个 IAM 用户提供访问权限。
- 要了解如何向第三方提供对您的资源的访问权限 AWS 账户，请参阅 IAM 用户指南中的向第三方提供访问权限。 AWS 账户
- 要了解如何通过身份联合验证提供访问权限，请参阅《IAM 用户指南》中的为经过外部身份验证的用户（身份联合验证）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 中的跨账户资源访问。

Amazon MQ 的合规性验证

Third-party 作为多项合规计划的一部分，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。有关您在使用时的合规责任的更多信息 AWS 服务，请参阅AWS 安全文档。

Amazon MQ 中的恢复能力

AWS全球基础架构围绕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失效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、容错能力和可扩展性。

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

作为一项托管服务，Amazon MQ 受到 AWS 全球网络安全的保护。有关 AWS 安全服务以及如何 AWS 保护基础设施的信息，请参阅AWS 云安全。要使用基础设施安全的最佳实践来设计您的 AWS 环境，请参阅 S AWS ecurity Pillar Well-Architected Fram ework 中的基础设施保护。

您可以使用 AWS 已发布的 API 调用通过网络访问 Amazon MQ。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（临时 Diffie-Hellman）或 ECDHE（临时椭圆曲线 Diffie-Hellman）。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

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。这大大降低了您的经纪人对来自公共互联网的分布式拒绝服务 (DDoS) 攻击的敏感性。有关更多信息，请参阅 AWS 安全博客上的如何通过减少攻击面来帮助为 DDo S 攻击做好准备。

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

为提高私有代理的安全性，您应通过正确配置 Amazon VPC 安全组来限制不必要协议和端口的连接。例如，要在允许访问 Web 控制台的同时限制对 OpenWire 大多数协议的访问，可以只允许访问 61617 和 8162。这会通过屏蔽您未使用的协议来限制您的曝光，同时允许 OpenWire 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：

- 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

监控是维护 AWS 解决方案可靠性、可用性和性能的重要组成部分。您应该从 AWS 解决方案的所有部分收集监控数据，以便在出现多点故障时可以更轻松地对其进行调试。 AWS 提供了多种用于监控您的 Amazon MQ 资源和响应潜在事件的工具：

您可以使用 CloudWatch 来查看和分析您的 Amazon MQ 经纪商的指标。您可以通过 CloudWatch 控制台、或，查看和分析您的经纪商指标 CloudWatch AWS CLI。 AWS CLI CloudWatch Amazon MQ 的指标会自动从经纪人那里进行轮询，然后推送到每分钟。 CloudWatch 对于 ActiveMQ 经纪商 CloudWatch ，仅监控前 1000 个目的地。对于 RabbitMQ 经纪商，仅 CloudWatch 监控前 500 个目的地，按消费者数量排序。

有关 Amazon MQ 指标的完整列表，请参阅Amazon MQ 适用于 ActiveMQ 经纪商的可用 CloudWatch 指标。

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。

访问亚马逊 MQ 的 CloudWatch 指标

您可以使用 AWS 管理控制台 AWS CLI、和 API 访问 CloudWatch 指标。

您可能希望在不使用的情况下访问 CloudWatch 指标 AWS 管理控制台。

要使用访问 Amazon MQ 指标 AWS CLI，请使用命令。`get-metric-statistics`有关更多信息，请参阅 Amazon CloudWatch 用户指南中的获取指标的统计数据。

要使用 CloudWatch API 访问亚马逊 MQ 指标，请使用操作。`GetMetricStatistics`有关更多信息，请参阅 Amazon CloudWatch 用户指南中的获取指标的统计数据。

使用访问 CloudWatch 指标 AWS 管理控制台

以下示例向您展示了如何使用访问亚马逊 MQ 的 CloudWatch 指标 AWS 管理控制台。如果您已经登录到 Amazon MQ 控制台，请在经纪人详情页面上，选择操作，查看指标。 CloudWatch

1. 登录 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。

访问 Prometheus 指标

> **Note**
>
> Prometheus 指标仅适用于 RabbitMQ 4.2 及更高版本。ActiveMQ 经纪商不支持 Prometheus 指标。

亚马逊 MQ 现在支持 RabbitMQ 经纪商的亚马逊 MQ 的 Prometheus 指标。Prometheus 指标使您能够将经纪商的可观察性集成到现有的监控基础架构中，从而使您可以统一了解经纪商绩效以及其他服务。借助 Prometheus 指标，您可以设置精细的警报和仪表板，以主动检测和响应消息传递工作负载中的问题。

从 RabbitMQ 4.2 开始，适用于 RabbitMQ 的亚马逊 MQ 支持 Prometheus 指标，允许您使用 Prometheus 监控系统获取经纪商指标。支持以下端点：

- `/metrics`
- `/metrics/detailed`
- `/metrics/memory-breakdown`

不支持该`/metrics/per-object`终端节点。

有关每个端点公开的指标的更多信息，请参阅 RabbitMQ 文档中的 Prometheus 指标。

Prometheus 指标与指标 CloudWatch

适用于 RabbitMQ 的 Amazon MQ 通过 Prometheus 终端节点和. CloudWatch 虽然两者都提供了对经纪人运行状况的可见性，但它们的范围和用法各不相同。

Prometheus 端点公开了一组关于 RabbitMQ 经纪人健康状况的更丰富的汇总指标，涵盖了更广泛的经纪商内部数据，例如连接流失、频道活动、队列和交易所统计数据以及 Raft 共识指标。它们适合与基于 Prometheus 的现有监控基础架构和细粒度警报集成。

CloudWatch 指标是从 Prometheus 端点获取的经纪商指标的精选子集。有关可用 CloudWatch 指标的完整列表，请参阅适用于 RabbitMQ 经纪商的亚马逊 MQ 可用 CloudWatch 指标。

在中 CloudWatch，指标始终以至少 60 秒的时间间隔进行聚合，然后再进行可视化。相比之下，Prometheus 公开了原始指标数据点，而像 Grafana 这样的仪表板解决方案默认情况下无需聚合即可对单个数据点进行可视化。因此，相同指标的可视化效果可能会在 Prometheus 和 CloudWatch Prometheus 之间存在差异，具体取决于中使用的统计数据 CloudWatch

> **Note**
>
> 我们建议使用 Prometheus 对亚马逊 MQ 的 RabbitMQ 运营指标进行非汇总监控。

获取和访问 Prometheus 端点

您可以使用或获取 Amazon MQ for RabbitMQ 代理的 Prometheus 终端节点。 AWS 管理控制台 AWS CLI

- AWS 管理控制台— 导航到亚马逊 MQ 控制台，打开您的经纪商的详细信息页面，然后在 “连接” 部分下找到 Prometheus 终端节点。
- AWS CLI— 使用以下`describe-broker`命令：

```
aws mq describe-broker --broker-id <broker-id>
```

Prometheus 端点将在下的响应中返回。`BrokerInstances.Endpoints`

适用于 RabbitMQ 的 Amazon MQ Prometheus 支持使用与代理相同的身份验证方案。有关支持的身份验证方法的更多信息，请参阅适用于 RabbitMQ 身份验证和授权的亚马逊 MQ。要了解如何在 Prometheus 中配置身份验证，请参阅 Prometheus 文档中的 http _config。

Prometheus 配置最佳实践

- 将抓取周期配置为 60 秒或更长。出于操作安全考虑，建议这样做。

抓取配置示例

以下各节提供了适用于 RabbitMQ 的亚马逊 MQ 的 Prometheus 抓取配置示例。<broker-prometheus-endpoint>替换为您的经纪商的 Prometheus 终端节点主机名，<username>并<password>替换为您的经纪人凭证。

建议的配置

对于大多数用例，建议使用以下配置。抓取/metrics终端节点可提供有关集群整体运行状况的精心汇总的指标，使您可以清楚地了解代理性能，而无需花费详细的指标收集开销。

```
global:
  scrape_interval: 60s

scrape_configs:
  - job_name: 'rabbitmq-aws-cluster'
    scheme: https
    basic_auth:
      username: <username>
      password: <password>
    metrics_path: '/metrics'
    static_configs:
      - targets:
        - '<broker-prometheus-endpoint>:16001'
        - '<broker-prometheus-endpoint>:16002'
        - '<broker-prometheus-endpoint>:16003'
```

详细的指标配置

以下配置收集了更多详细的指标系列，以便更深入地观察特定的经纪商组件。

```
global:
  scrape_interval: 60s

scrape_configs:
  - job_name: 'rabbitmq-connection-churn'
    scheme: https
```

```
    basic_auth:
      username: <username>
      password: <password>
    metrics_path: '/metrics/detailed'
    params:
      family: ['connection_churn_metrics']
    static_configs:
      - targets:
        - '<broker-prometheus-endpoint>:16001'
        - '<broker-prometheus-endpoint>:16002'
        - '<broker-prometheus-endpoint>:16003'
  - job_name: 'rabbitmq-ra'
    scheme: https
    basic_auth:
      username: <username>
      password: <password>
    metrics_path: '/metrics/detailed'
    params:
      family: ['ra_metrics']
    static_configs:
      - targets:
        - '<broker-prometheus-endpoint>:16001'
        - '<broker-prometheus-endpoint>:16002'
        - '<broker-prometheus-endpoint>:16003'
  - job_name: 'rabbitmq-queue'
    scheme: https
    basic_auth:
      username: <username>
      password: <password>
    metrics_path: '/metrics/detailed'
    params:
      family: ['queue_metrics']
    static_configs:
      - targets:
        - '<broker-prometheus-endpoint>:16001'
        - '<broker-prometheus-endpoint>:16002'
        - '<broker-prometheus-endpoint>:16003'
  - job_name: 'rabbitmq-exchange'
    scheme: https
    basic_auth:
      username: <username>
      password: <password>
    metrics_path: '/metrics/detailed'
    params:
```

```
      family: ['exchange_metrics']
    static_configs:
      - targets:
        - '<broker-prometheus-endpoint>:16001'
        - '<broker-prometheus-endpoint>:16002'
        - '<broker-prometheus-endpoint>:16003'
  - job_name: 'rabbitmq-connection'
    scheme: https
    basic_auth:
      username: <username>
      password: <password>
    metrics_path: '/metrics/detailed'
    params:
      family: ['connection_metrics']
    static_configs:
      - targets:
        - '<broker-prometheus-endpoint>:16001'
        - '<broker-prometheus-endpoint>:16002'
        - '<broker-prometheus-endpoint>:16003'
  - job_name: 'rabbitmq-channel'
    scheme: https
    basic_auth:
      username: <username>
      password: <password>
    metrics_path: '/metrics/detailed'
    params:
      family: ['channel_metrics']
    static_configs:
      - targets:
        - '<broker-prometheus-endpoint>:16001'
        - '<broker-prometheus-endpoint>:16002'
        - '<broker-prometheus-endpoint>:16003'
  - job_name: 'rabbitmq-exchange-count'
    scheme: https
    basic_auth:
      username: <username>
      password: <password>
    metrics_path: '/metrics/detailed'
    params:
      family: ['exchange_names']
    static_configs:
      - targets:
        - '<broker-prometheus-endpoint>:16001'
        - '<broker-prometheus-endpoint>:16002'
```

```
      - '<broker-prometheus-endpoint>:16003'
```

Amazon MQ 适用于 ActiveMQ 经纪商的可用 CloudWatch 指标

Amazon MQ for ActiveMQ 指标

指标	单位	说明
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。有关突发余额在 Amazon EBS 中的运作方式的更多信息，请参阅：I/O积分和突发性能。
CpuCreditBalance	积分 (vCPU-minutes)	**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。

指标	单位	说明
		在获得积分后，积分将在积分余额中累积；在花费积分后，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。

指标	单位	说明
NetworkConnectorConnectionCount	计数	使用经纪商网络中连接到经纪商的节点数量 NetworkConnector。
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	您可以使用的最大客户端数量与您的经纪商建立连接 OpenWire。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。

指标	单位	说明
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。
WsMaximumConnections	计数	您可以使用的最大客户端数量与您的经纪商建立连接 WebSocket。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	说明
Broker	代理的名称 **Note** 单实例代理具有后缀 -1。高可用性 active/standby 代理的冗余对具有后缀 -1 和 -2。

ActiveMQ 目标（队列和主题）指标

Important

以下指标包括 CloudWatch 轮询期间的每分钟计数。

- EnqueueCount
- ExpiredCount
- DequeueCount

- DispatchCount
- InFlightCount

例如，在五分钟CloudWatch 的时间段内，EnqueueCount有五个计数值，每个值代表该时段的一分钟部分。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	单位	说明
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。

指标	单位	说明
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。
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	说明
Broker	代理的名称。

维度	说明
	Note 单实例代理具有后缀 -1。用于实现高可用性的 active/standby 代理具有后缀-1和冗-2余对。
Topic 或 Queue	主题或队列的名称。
NetworkConnector	网络连接器的名称。

适用于 RabbitMQ 经纪商的亚马逊 MQ 可用 CloudWatch 指标

Warning

从 RabbitMQ 4.2 开始，已被弃用RabbitMQIOReadAverageTime，不会发布RabbitMQIOWriteAverageTime有意义的值。这些指标将在下一个主要的 RabbitMQ 版本 CloudWatch 中删除。

RabbitMQ 代理指标

Note

不建议将管理插件用于开源 RabbitMQ 的生产或长期监控。我们建议使用 Prometheus 来查询 RabbitMQ 4.2 及更高版本的每节点指标。

指标	单位	说明
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。

指标	单位	说明
ChannelCount	计数	在代理上建立的通道总数。 **Important** 频道的概念特定于 AMQP 0-9-1。
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。 **Important** 此指标仅反映 AMQP 0-9-1 协议活动。有关 AMQP 1.0 的指标，请参阅。访问 Prometheus 指标

指标	单位	说明
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。 **Important** 此指标仅反映 AMQP 0-9-1 协议活动。有关 AMQP 1.0 的指标，请参阅。访问 Prometheus 指标
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。 **Important** 此指标仅反映 AMQP 0-9-1 协议活动。有关 AMQP 1.0 的指标，请参阅。访问 Prometheus 指标
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。

指标	单位	说明
RabbitMQMemLimit	字节	RabbitMQ 代理的 RAM 限制。该指标因实例类型而异。有关更多信息，请参阅 the section called “内存和磁盘警报”。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。
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。该指标因实例类型和部署模式而异。有关更多信息，请参阅 the section called “内存和磁盘警报”。
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。

指标	单位	说明
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	说明
Broker	代理的名称。

RabbitMQ 节点指标

指标	单位	说明
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。该指标因实例类型而异。有关更多信息，请参阅 the section called “内存和磁盘警报”。
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。

指标	单位	说明
RabbitMQDiskFreeLimit	字节	RabbitMQ 节点的磁盘限制。该指标因实例类型和部署模式而异。有关更多信息，请参阅 the section called “内存和磁盘警报”。
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。
ExchangeCount	计数	节点上配置的交换总数。适用于 RabbitMQ 4.2 及更高版本。
QueueCount	计数	节点上配置的队列总数。适用于 RabbitMQ 4.2 及更高版本。
ConnectionCount	计数	节点上建立的连接总数。适用于 RabbitMQ 4.2 及更高版本。
ChannelCount	计数	节点上建立的信道总数。适用于 RabbitMQ 4.2 及更高版本。
ConsumerCount	计数	连接到该节点的消费者总数。适用于 RabbitMQ 4.2 及更高版本。
MessageCount	计数	节点上队列中的消息总数。适用于 RabbitMQ 4.2 及更高版本。

指标	单位	说明
MessageReadyCount	计数	节点上队列中的就绪消息总数。适用于 RabbitMQ 4.2 及更高版本。
MessageUnacknowledgedCount	计数	节点上队列中未确认的消息总数。适用于 RabbitMQ 4.2 及更高版本。

从 RabbitMQ 节点指标中聚合集群范围的指标

要获取集群范围的汇总指标，您可以通过筛选代理名称和指标名称，在 CloudWatch 控制台上找到相应的每节点指标。然后，通过单击复选框来选择这些指标，然后选择添加数学 > 常用 > 总和。

RabbitMQ 节点指标的维度

维度	说明
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	单位	说明
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	说明
Queue	队列的名称。
VirtualHost	虚拟主机的名称。
Broker	代理的名称。

RabbitMQ 网络指标

指标	单位	说明
NetworkOut	字节	实例在所有网络接口上发送的字节数。此指标用于确定来自单个实例的传出网络流量。报告的数字是该时间段内发送的字节数。如果您使用的是基本（5 分钟）监视并且统计数据为 Sum，则可以将此数字除以 300 进行查找 Bytes/second。如果您使用的是详细（1 分钟）监控且统计数据为 Sum，请将其除以 60。您也可以使用 CloudWatch 公制数学函数DIFF_TIME 来查找每秒的字节数。例如，如果您将图表 NetworkOut 设置 CloudWatch 为m1，则指标数学公式将m1/(DIFF_TIME(m1)) 返回该指标。 bytes/second有关DIFF_TIME 和其他指标数学函数的更多信息，请参阅使用指标数学。 有意义的统计量：总和、平均值、最小值、最大值
NetworkIn	字节	实例在所有网络接口上收到的字节数。此指标用于确定流向单个实例的传入网络流量。报告的数量是该期间内接收的字节数。如果您使用的是基本（5 分钟）监视并且统计数据为 Sum，则可以将此数字除以 300 进行查找 Bytes/second。如果您使用的是详细（1 分钟）监控且统计数据为 Sum，请将其除以 60。您也可以使用 CloudWatch 公制数学函数DIFF_TIME 来查找每秒的字节数。例如，如果您将图表 NetworkIn 设置 CloudWatch 为m1，则指标数学公式将m1/(DIFF_TIME(m1)) 返回该指标。 bytes/second有关DIFF_TIME 和其他指标数学函数的更多信息，请参阅使用指标数学。 有意义的统计量：总和、平均值、最小值、最大值

RabbitMQ 代理的维度

维度	说明
Broker	代理的名称。

配置 Amazon MQ for RabbitMQ 日志

当您为 RabbitMQ 代理启用 CloudWatch 日志记录功能时，Amazon MQ 会使用服务相关角色向其发布一般日志。 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。所有后续的 RabbitMQ 代理都将使用相同的服务相关角色向其发布日志。 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 “使用服务关联角色”。

使用记录亚马逊 MQ API 调用 AWS CloudTrail

Amazon MQ 与一项服务集成 AWS CloudTrail，该服务提供用户、角色或服务发起的 Amazon MQ 调用的记录。 AWS CloudTrail 捕获与亚马逊 MQ 代理和配置相关的 API 调用作为事件，包括来自亚马逊 MQ 控制台的调用和来自亚马逊 MQ API 的代码调用。有关的更多信息 CloudTrail，请参阅《AWS CloudTrail 用户指南》。

> **Note**
>
> CloudTrail 不记录与 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，您可以将 Amazon MQ 配置为向亚马逊日志发布一般日志和审核日志。 CloudWatch

使用 CloudTrail 收集到的信息，您可以识别对 Amazon MQ API 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，则可以将 CloudTrail 事件持续传输到 Amazon S3 存储桶。如果您未配置跟踪，则可以在 CloudTrail 控制台中查看事件历史记录中的最新事件。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。

中的亚马逊 MQ 信息 CloudTrail

当您创建 AWS 账户时，已启 CloudTrail 用。当支持的 Amazon MQ 事件活动发生时，会将其与其他 AWS 服务 CloudTrail 事件一起记录在事件历史记录中。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。

跟踪允许 CloudTrail 将日志文件传输到 Amazon S3 存储桶。您可以创建跟踪，以便在您的 AWS 账户中持续记录事件。默认情况下，当您使用创建跟踪时 AWS 管理控制台，该跟踪将应用于所有 AWS 区域。该跟踪记录来自所有 AWS 区域的事件，并将日志文件传输到指定的 Amazon S3 存储桶。您还可以配置其他 AWS 服务，以进一步分析和处理 CloudTrail 日志中收集的事件数据。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
- 配置 Amazon SNS 通知 CloudTrail
- 接收来自多个区域的 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**
>
> RebootBroker 当您重新启动代理时，日志文件会被记录下来。在维护时段内，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？
- 请求是由其他 AWS 服务机构发出的？

有关更多信息，请参阅《CloudTrail用户指南》中的 “AWS CloudTrail 用户身份元素”。

示例 Amazon MQ 日志文件条目

跟踪是一种配置，允许将事件作为日志文件传输到指定的 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。

以下示例显示了 CreateBrokerAPI 调用的 CloudTrail 日志条目。

> **Note**
>
> 由于 CloudTrail 日志文件不是公共 API 的有序堆栈跟踪，因此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 for ActiveMQ 日志

要允许 Amazon MQ 将日志发布到 CloudWatch 日志，您必须向您的 Amazon MQ 用户添加权限，并在创建或重启代理之前为 Amazon MQ 配置基于资源的策略。

> **Note**
>
> 当您打开日志并从 ActiveMQ Web 控制台发布消息时，消息内容将发送 CloudWatch 到日志并显示在日志中。

以下介绍为 ActiveMQ 代理配置 CloudWatch 日志的步骤。

主题

了解 CloudWatch 日志中登录的结构

您可以在创建代理时配置高级代理设置，或在编辑代理时启用常规和审计日志记录。

常规日志记录启用默认INFO日志级别（不支持DEBUG日志记录），并发布activemq.log到您 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
```

审核日志允许记录使用 JMX 或 ActiveMQ Web 控制台执行的管理操作，并将其audit.log发布到您账户中的日志组。 CloudWatch 日志组的格式如下所示：

```
/aws/amazonmq/broker/b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9/audit
```

根据您使用的是单实例代理还是代理，Amazon MQ 会在每个日志组中创建一个或两个日志流。active/standby 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 日志用户指南中的使用日志组和 CloudWatch 日志流。

向 Amazon MQ 用户添加 **CreateLogGroup** 权限

要允许 Amazon MQ 创建 CloudWatch 日志组，您必须确保创建或重启代理的用户拥有该权限。logs:CreateLogGroup

> **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IAM-based 策略向附加了此策略`logs:CreateLogGroup`的用户授予权限。

JSON

```
{
                    "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 日志 API 参考*CreateLogGroup*中的。

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。

以下基于资源的策略授予对logs:CreateLogStream和的权限logs:PutLogEvents。 AWS

JSON

```
{
                          "Version":"2012-10-17",
                          "Statement": [
                              {
                                  "Effect": "Allow",
                                  "Principal": { "Service":
 "mq.amazonaws.com" },
                                  "Action": [ "logs:CreateLogStream",
 "logs:PutLogEvents" ],
                                  "Resource": "arn:aws:logs:*:*:log-group:/aws/
amazonmq/*"
                              }
                          ]
                      }
```

必须使用配置此基于资源的策略， AWS CLI 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，因此您只需为每个 AWS 账户、每个区域配置一次基于资源的策略。

Cross-service 混乱的副手预防

混淆代理问题是一个安全性问题，即不具有某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。在中 AWS，跨服务模仿可能会导致混乱的副手问题。 Cross-service 当一个服务（调用服务）调用另一个服务（被调用的服务）时，可能会发生模仿行为。可以操纵调用服务以使用其权限对另一个客户的资源进行操作，否则该服务不应有访问权限。为了防止这种情况，我们 AWS 提供了一些工具，帮助您保护所有服务的数据，这些服务委托人已被授予访问您账户中资源的权限。

我们建议在基于资源的 Amazon MQ 策略中使用aws:SourceArn和aws:SourceAccount全局条件上下文密钥来限制一个或多个指定代理访问 CloudWatch 日志。

> **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。

以下示例演示了一种基于资源的策略，该策略将 CloudWatch 日志访问权限限制为单个 Amazon MQ 代理。

JSON

```
{
                    "Version":"2012-10-17",
                    "Statement": [
                        {
                        "Effect": "Allow",
                        "Principal": {
                            "Service": "mq.amazonaws.com"
                        },
                        "Action": [
                            "logs:CreateLogStream",
                            "logs:PutLogEvents"
```

```
                            ],
                            "Resource": "arn:aws:logs:*:*:log-group:/aws/
amazonmq/*",
                            "Condition": {
                                "StringEquals": {
                                "aws:SourceAccount": "123456789012",
                                "aws:SourceArn": "arn:aws:mq:us-
west-1:123456789012:broker:my-broker:123456789012"
                                }
                            }
                            }
                        ]
                        }
```

您还可以将基于资源的策略配置为限制账户中所有经纪人的 CloudWatch 日志访问权限，如下所示。

JSON

```
{
                            "Version":"2012-10-17",
                            "Statement": [
                            {
                                "Effect": "Allow",
                                "Principal": {
                                "Service": [
                                    "mq.amazonaws.com"
                                ]
                                },
                                "Action": [
                                "logs:CreateLogStream",
                                "logs:PutLogEvents"
                                ],
                                "Resource": "arn:aws:logs:*:*:log-group:/aws/
amazonmq/*",
                                "Condition": {
                                "ArnLike": {
                                    "aws:SourceArn":
 "arn:aws:mq:*:123456789012:broker:*"
                                },
                                "StringEquals": {
                                    "aws:SourceAccount": "123456789012"
                                }
```

```
                }
            }
            ]
        }
```

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。

使用 Amazon MQ 对 CloudWatch 日志配置进行故障排除

在某些情况下，CloudWatch 日志可能并不总是按预期运行。此部分概述了常见问题并说明如何解决这些问题。

日志组未显示在 CloudWatch

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。

日志流不出现在 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。

Amazon MQ 中的配额

本主题列出了 Amazon MQ 中的限制。对于特定 AWS 账户，可以更改以下许多限制。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

主题

代理

下表列出了与 Amazon MQ 代理相关的配额。

限制	说明
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	200
小型代理每个协议的线级连接数	**Important** 不适用于 RabbitMQ 代理。

限制	说明
	`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
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 配置相关的配额。

限制	说明
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

Users

下表列出了与 Amazon MQ ActiveMQ 代理用户相关的配额。

限制	说明
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 数据存储相关的配额。

限制	说明
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。

限制	说明
每个大型代理的存储容量	`mq.m5.*` 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	**Important** 不适用于 RabbitMQ 代理。 `mq.m5.*` 实例类型代理为 50GB。

API 限制

为了保持服务带宽，在所有 Amazon APIs MQ 中，每个 AWS 账户汇总了以下限制配额。有关亚马逊 MQ 的更多信息 APIs，请参阅亚马逊 MQ REST API 参考。

> **Important**
>
> 这些配额不适用于适用于 ActiveMQ 的亚马逊 MQ 或适用于 RabbitMQ 代理消息的亚马逊 MQ。 APIs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。有关一般故障排除，请参阅the section called “问题排查：一般 Amazon MQ 问题”。要排查特定引擎版本的问题，请参阅以下部分。

Amazon MQ 上的 ActiveMQ 问题排查

故障排除主题	说明
一般故障排除	使用本节信息帮助您诊断和解决使用 Amazon MQ 代理上的 ActiveMQ 时可能遇到的常见问题。
BROKER_ENI_已删除	当您删除代理的弹性网络接口（ENI）时，Amazon MQ 上的 ActiveMQ 将引发 `BROKER_ENI_DELETED` 警报。
经纪人_OOM	当代理因内存容量不足而进入重启循环时，Amazon MQ 上的 ActiveMQ 将引发 BROKER_OOM 警报。

Amazon MQ 上的 RabbitMQ 问题排查

故障排除主题	说明
一般故障排除	诊断使用 RabbitMQ 代理时可能遇到的常见问题。
RABBITMQ_MEMORY_ALARM	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`

故障排除主题	说明
RABBITMQ_INVALID_KMS_KEY	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，亚马逊 MQ 上的 RabbitMQ 将引发一个 INVALID_KMS_KEY 关键操作所需的代码。 AWS Key Management Service
RABBITMQ_INVALID_ASSUMEROLE	当亚马逊 MQ 无法担任中指定的 IAM 角色 ARN 时，亚马逊 MQ 上的 RabbitMQ 将引发一个 INVALID_ASSUMEROLE 关键操作所需的代码。`aws.arns.assume_role_arn`
RabbitMQ_INVALID_ARN_LDAP	当 LDAP 服务账户密码 ARN 无效或无法访问时，亚马逊 MQ 上的 RabbitMQ 将引发一个 INVALID_ARN_LDAP 关键操作所需的代码。
RABBITMQ_INVALID_ARN_HTTP	当 HTTP auth_backend 的一个或多个 SSL 证书或 HTTP auth_backend 密钥文件的 ARN 无效或无法访问时，亚马逊 MQ 上的 RabbitMQ 将引发一个 INVALID_ARN_HTTP 关键操作必需代码。

故障排除主题	说明
RabbitMQ_INVALID_ARN_SSL	当外部身份验证机制的 CA 证书信任库的一个或多个 ARN 无效或无法访问时，亚马逊 MQ 上的 RabbitMQ 将引发 INVALID_ARN_SSL 关键操作所需的代码。
RABBITMQ_INVALID_ARN	当代理配置中的一个或多个 ARN 无效或无法访问时，亚马逊 MQ 上的 RabbitMQ 将引发一个 INVALID_ARN 关键操作所需的代码。
RabbitMQ_disk_alarm	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。
RABBITMQ_BROKER_NOT_UPGRADEABLE_TO_V4	在启用了经典队列或 Khepri 的代理上尝试升级到 RabbitMQ 4 时，亚马逊 MQ 上的 RabbitMQ 将引发 RABBITMQ_BROKER_NOT_UPGRADEABLE_TO_V4 关键操作必填代码。

问题排查：一般 Amazon MQ 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

- 我创建了一个代理，但代理创建失败。
- 我的代理重启，我不知道原因是什么。

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 网络的更多信息，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分组，因为 Amazon MQ 上的 ActiveMQ 和 Amazon MQ 上的 RabbitMQ 使用不同的连接端口。

   Amazon MQ 上的 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 上的 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。

ActiveMQ on Amazon MQ

测试 Amazon MQ 上的 ActiveMQ 代理的网络连接性

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active/standby部署，请同时测试活动端点和备用端点。 active/standby 端点用后缀标识，`-1`或者`-2`添加到唯一的经纪商 ID 中。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。根据8162需要*port*替换为 Web 控制台的端口号或其他线级端口以测试其他协议。

   > **Note**
   >
   > 对于 active/standby 部署，如果您使用备用终端节点运行`telnet`，则会收到一条`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最大Wire-level 连接限制。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>
```

```
        ...
```

RabbitMQ on Amazon MQ

测试 Amazon MQ 上的 RabbitMQ 代理的网络连接性

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*port*替换为 Web 控制台443的端口，也可以替换5671为线级 AMQP 连接的端口。

```
$ 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` 命令执行成功，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**
     >
     > 此步骤不适用于具有公共访问权限的 Amazon MQ 上的 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，但是我的客户无法连接并返回 SSL 异常。

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。有关代理维护窗口的更多信息，请参阅 the section called “计划代理维护”。
- 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.t3.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 上的 ActiveMQ – 检查 `CpuUtilization` 以了解代理当前使用的已分配 Amazon EC2 compute unit 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 上的 RabbitMQ – 检查 `SystemCpuUtilization` 以了解代理当前使用的已分配 Amazon EC2 compute unit 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正确实例类型的更多信息，请参阅 Broker instance types。

Amazon MQ 上的 ActiveMQ 问题排查

使用本节信息帮助您诊断和解决使用 Amazon MQ 代理上的 ActiveMQ 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。

有关在 Amazon MQ 上配置 ActiveMQ 以将日志 CloudWatch 发布到日志的更多信息，请参阅配置日志。

在代理重启或维护窗口后，即使状态为 RUNNING，我也无法连接到我的经纪#。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 经纪商的以下指标：

- **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” 中查看您在 Amazon MQ 上的 ActiveMQ 代理的通用经纪人日志。 CloudWatch 如果已达到限制，您会在代理日志中看到 `Reached Maximum Connections`。有关亚马逊 MQ 代理上的 ActiveMQ CloudWatch 日志的更多信息，请参阅。the section called “了解 CloudWatch 日志中登录的结构”

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。

我看到了异常 org.apache.jasper# JasperException: ####### ActiveMQ ###### JSP #######。

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 上的 RabbitMQ

使用本节信息帮助您诊断和解决使用 Amazon MQ 代理上的 RabbitMQ 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。

如果您无法在中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。

如何在 Amazon MQ 上的 RabbitMQ 中启用插件？

Amazon MQ 上的 RabbitMQ 目前仅支持 RabbitMQ 管理、Shovel、Federation、一致性哈希交换插件，这些插件默认启用。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。

集群部署已暂停我的队列同步。

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”。

我的 Amazon MQ for RabbitMQ 单实例代理处于重启循环中。

引发高内存警报的 Amazon MQ for RabbitMQ 单实例代理，如果在重启时没有足够内存启动，将面临不可用的风险。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。如果您的代理处于重启循环中，您将无法应用 Amazon MQ 推荐的最佳实践来解决高内存警报。

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。与集群部署不同，您可以在单实例代理出现高内存警报时升级它，因为在重启期间无需在节点之间执行队列同步。

我无法访问我的经纪商的所有管理员账户。

您可以使用 IAM 身份验证恢复访问权限。为您的 AWS 账户启用出站 Web 联合身份验证，创建有权获取 Web 身份令牌的 IAM 角色，将您的代理配置为通过 OAuth 2.0 接受 IAM 身份验证，然后使用 IAM

证书获取 JWT 令牌并创建新的管理员用户。有关详细说明，请参阅the section called “使用 IAM 身份验证和授权”。

Amazon MQ 上的 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 上的 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 上的 ActiveMQ：代理内存不足警报

当代理因内存容量不足而进入重启循环时，Amazon MQ 上的 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。	

亚马逊 MQ 指标 CloudWatch	内存使用过高的原因	
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 上的 ActiveMQ 最佳实践中了解更多关于确保有效性能的信息。

Amazon MQ for RabbitMQ：高内存警报

当代理的内存使用量（按 CloudWatch 指标RabbitMQMemUsed标识）超过由标识的内存限制时，适用于 RabbitMQ 的 Amazon MQ 将发出内存不足警报。RabbitMQMemLimit

引发高内存警报的 RabbitMQ 代理将阻止所有发布消息的客户端。您的代理可能会进入重启循环，经历队列同步暂停，或出现其他使警报诊断和解决复杂化的问题。

要诊断和解决高内存警报，请首先遵循所有适用于 RabbitMQ 的最佳实践，然后完成以下步骤。

> **Important**
>
> - RabbitMQMemLimit 由 Amazon MQ 设置，并针对每个主机实例类型的可用内存进行了专门调整。有关更多信息，请参阅 the section called “内存和磁盘警报”。
> - 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。

步骤 1：诊断高内存警报

有两种方法可以诊断 Amazon MQ for RabbitMQ 代理上的高内存警报。我们建议您在中同时查看 RabbitMQ 网页控制台和亚马逊 MQ 指标。 CloudWatch

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 管理控制台 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。有关通过 RabbitMQ Web 控制台获取的内存使用量详细信息，请参阅 RabbitMQ 服务器文档网站上的内存使用分析。

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。
`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。

亚马逊 MQ 指标 CloudWatch	内存使用过高的原因	
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。	

步骤 2：处理和防止高内存警报

> **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。

遵循所有 RabbitMQ 的最佳实践作为常规预防方法。对于您识别的每个具体影响因素，我们推荐采取以下系列措施来应对和预防 RabbitMQ 高内存警报。

高内存使用来源	Amazon MQ 处理建议	Amazon MQ 预防建议
消息数量	消费发布到队列的消息、清空队列中的消息或从代理中删除队列。	启用惰性队列，并设置或降低队列深度限制。
队列数	减少队列数。	设置或降低队列数量限制。

高内存使用来源	Amazon MQ 处理建议	Amazon MQ 预防建议
连接数	减少连接数。	设置或降低连接数限制。
通道编号	减少频道数量。	设置客户端应用程序上的每个连接的最大通道数。
使用者数	减少连接到代理的使用者数。	设置一个小的使用者预提取限制。
消息发布速率	降低发布商向代理发送消息的速率。	开启发布者确认。
客户端连接尝试速率	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。	使用长期连接来降低连接尝试的数量和频率。

代理内存警报解决后，您可以将主机实例类型升级到具有更多资源的实例。有关如何更新代理实例类型的信息，请参阅《Amazon MQ REST API 参考》中的 `UpdateBrokerInput`。

> **Note**
>
> 您不能将代理从 `mq.m5.x` 实例类型降级到 `mq.t3.micro` 实例类型。要降级，必须删除代理并创建新代理。

亚马逊 MQ 上的 RabbitMQ：无效 AWS Key Management Service Key

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，亚马逊 MQ 上的 RabbitMQ 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 上的 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。有关更多信息，请参阅 the section called “内存和磁盘警报”。

引发磁盘限制警报的 Amazon MQ for RabbitMQ 代理将无法发布新消息。如果发布者和使用者处于同一连接上，则使用者也将无法接收消息。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：

- 创建新连接以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`。我们还建议让发布者和使用者使用不同的连接。

Amazon MQ for RabbitMQ：实例类型变更警报

`RABBITMQ_CLUSTER_DISK_USAGE_TOO_HIGH_FOR_INSTANCE_CHANGE` 表示由于当前 RabbitMQ 节点磁盘使用率过高，无法继续进行请求的代理实例类型变更。当当前磁盘使用量超过指标所识别的请求实例类型的可用空间时，适用于 RabbitMQ 的 Amazon MQ 将发出此警报。 CloudWatch `RabbitMQDiskFree`

进入 RABBITMQ_CLUSTER_DISK_USAGE_TOO_HIGH_FOR_INSTANCE_CHANGE 状态的 RabbitMQ 代理将继续对您的应用程序可用，但请求的实例类型变更将不会进行。Amazon MQ 允许在此状态下重启代理，但只要磁盘使用率仍高于请求实例类型的阈值，您就无法更改实例类型。在此状态下，代理将对尝试更改实例类型的 ModifyBroker API 操作返回异常。

诊断和处理实例类型变更警报

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。 MessageCount并且可以使用RabbitMQDiskFree指标进行诊断RABBITMQ_CLUSTER_DISK_USAGE_TOO_HIGH_FOR_INSTANCE_CHANGE。

要解除隔离状态并允许实例类型变更继续进行，请使用 Amazon MQ 管理控制台执行以下操作：

- 创建新连接以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**
>
> 在您采取必要措施
> 后，RABBITMQ_CLUSTER_DISK_USAGE_TOO_HIGH_FOR_INSTANCE_CHANGE 状态可能需要几个小时才能清除。

亚马逊 MQ 上的 RabbitMQ：IAM 代入角色无效

当中指定的 IAM 角色 ARN 无效或无法由亚马逊 MQ 担任时，亚马逊 MQ 上的 RabbitMQ 将引发一个 INVALID_ASSUMEROLE 关键操作所需的代码。aws.arns.assume_role_arn当该角色不存在、与经纪人位于不同的 AWS 账户中，或者与 mq.amazonaws.com 缺乏必要的信任关系时，就会发生这种情况。

RABBITMQ_INVALID_ASSUMEROLE 隔离区中的代理无法检索 LDAP 身份验证所需的凭据或证书，从而导致 LDAP 身份验证不可用。如果 LDAP 是唯一配置的身份验证方法，则用户将无法连接到代理。Amazon MQ 需要使用 IAM 角色才能访问代理配置中 ARN 引用的 AWS 资源，例如用于 LDAP 身份验证的 AWS Secrets Manager 密钥或 Amazon S3 对象。

诊断和解决 RABBITMQ_INVALID_ASSUMEROLE

要诊断和解决 RABBITMQ_INVALID_ASSUMEROLE 操作所需的代码，您必须使用亚马逊日志和控制台。 CloudWatch AWS Identity and Access Management

解决无效的代入角色问题

1. 导航至 Amazon CloudWatch Logs Insights，然后对您的经纪人的日志组运行以下查询`/aws/amazonmq/broker/<broker-id>/general`：

```
fields @timestamp, @message
| sort @timestamp desc
| filter @message like /error.*aws_arn_config/
| limit 10000
```

2. 查找类似于以下内容的错误消息：

```
[error] <0.254.0> aws_arn_config: {handle_assume_role,{error,
{assume_role_failed,"AWS service is unavailable"}}}
```

3. 检查 IAM 角色配置并修复任何问题，例如：
   - 确保该角色与经纪人位于同一个 AWS 账户中
   - 验证信任策略是否允许 mq.amazonaws.com 担任该角色
   - 确认该角色具有访问所需 AWS 资源的相应权限
4. 更新代理配置并重启代理。

亚马逊 MQ 上的 RabbitMQ：LDAP ARN 无效

当为 LDAP 服务账户密码配置的 ARN 无效或无法访问时，亚马逊 MQ 上的 RabbitMQ 将引发一个 INVALID_ARN_LDAP 关键操作所需的代码。这适用于`aws.arns.auth_ldap.dn_lookup_bind.password`或中指定的 ARN`aws.arns.auth_ldap.other_bind.password`，它必须引用包含纯文本 AWS Secrets Manager 密码的密钥。

RABBITMQ_INVALID_ARN_LDAP 隔离区中的代理无法使用 LDAP 服务账户进行身份验证，这导致 LDAP 身份验证不可用。如果 LDAP 是唯一配置的身份验证方法，则用户将无法连接到代理。ARN 无效可能是由格式错误的 ARN 语法、对不存在的密钥的引用、机密位于与代理不同的 AWS 区域或 secretsmanager: IAM 角色中的权限不足造成的。GetSecretValue

诊断并解决 RABBITMQ_INVALID_ARN_LDAP

要诊断和解决 RABBITMQ_INVALID_ARN_LDAP 操作所需的代码，您必须使用亚马逊日志和控制台。CloudWatch

解决无效的 LDAP ARN 问题

1. 导航至 Amazon CloudWatch Logs Insights，然后对您的经纪人的日志组运行以下查询`/aws/amazonmq/broker/<broker-id>/general`：

```
fields @timestamp, @message
| sort @timestamp desc
| filter @message like /error.*aws_arn_config/
| limit 10000
```

2. 查找类似于以下内容的错误消息：

```
[error] <0.254.0> aws_arn_config: {<<"could not resolve
 ARN 'arn:aws:secretsmanager:xxx' for configuration
 'aws.arns.auth_ldap.dn_lookup_bind.password', error: \"AWS service is unavailable
\"">>,{error,"AWS service is unavailable"}}
```

3. 检查 Secrets Manager 密钥并修复所有问题，例如：
   - 验证密钥是否与经纪人位于同一 AWS 区域
   - 确认 ARN 语法正确
   - 确保 IAM 角色具有 secretsmanager: 权限 GetSecretValue
4. 更新代理配置并重启代理。

亚马逊 MQ 上的 RabbitMQ：HTTP ARN 无效

当 HTTP auth_backend 的一个或多个 SSL 证书或 HTTP auth_backend 密钥文件的 ARN 无效或无法访问时，亚马逊 MQ 上的 RabbitMQ 将引发一个 INVALID_ARN_HTTP 关键操作必需代码。这适用于`aws.arns.auth_http.ssl_options.certfile`或中指定的 ARNS `aws.arns.auth_http.ssl_options.cacertfile`和`aws.arns.auth_http.ssl_options.keyfile`，它必须引用包含证书和私钥的 Amazon S3 对象和 AWS Secrets Manager 机密。

RABBITMQ_INVALID_ARN_HTTP 隔离区中的代理无法通过 HTTP 服务器进行身份验证。如果 HTTP 是唯一配置的身份验证方法，则用户将无法连接到代理。ARN 无效可能是由格式错误的 ARN 语法、对不存在的密钥的引用、机密位于与代理不同的 AWS 区域或 IAM 角色中的 s3: GetObject/secretsmanager:GetSecretValue 权限不足造成的。

诊断和解决 RABBITMQ_INVALID_ARN_HTTP

要诊断和解决 RABBITMQ_INVALID_ARN_HTTP 操作所需的代码，您必须使用亚马逊日志和控制台。CloudWatch

解决无效的 HTTP ARN 问题

1. 导航至 Amazon CloudWatch Logs Insights，然后对您的经纪人的日志组运行以下查询`/aws/amazonmq/broker/<broker-id>/general`：

```
fields @timestamp, @message
| sort @timestamp desc
| filter @message like /error.*aws_arn_config/
| limit 10000
```

2. 查找类似于以下内容的错误消息：

```
[error] <0.209.0> aws_arn_config: {<<"could not resolve ARN 'arn:aws:s3:::xxxx' for
 configuration 'aws.arns.auth_http.ssl_options.certfile', error: \"AWS service is
 unavailable\"">>,{error,"AWS service is unavailable"}}
```

3. 检查 S3 Object/Secrets 管理器密钥并修复所有问题，例如：

   - 验证资源是否与代理位于同一 AWS 区域

- 确认 ARN 语法正确
- 确保 IAM 角色具有 s3: GetObject 和 secretsmanager: 权限 GetSecretValue

4. 更新代理配置并重启代理。

亚马逊 MQ 上的 RabbitMQ：SSL ARN 无效

当外部身份验证机制的 CA 证书信任库的一个或多个 ARN 无效或无法访问时，亚马逊 MQ 上的 RabbitMQ 将引发 INVALID_ARN_SSL 关键操作所需的代码。这适用于`aws.arns.ssl_options.cacertfile`或中指定的 ARN`aws.arns.management.ssl.cacertfile`，它必须引用包含证书的 Amazon S3 或 ACM PCA 对象。

由于未配置有效的信任库，RABBITMQ_INVALID_ARN_SSL 隔离区中的代理无法在双向 TLS 握手期间对客户端证书进行身份验证。如果外部身份验证机制是唯一配置的身份验证方法，则用户将无法连接到代理。ARN 无效可能是由于 ARN 语法不正确、引用不存在的 S3 对象、S3 对象位于与代理不同的 AWS 区域或 IAM 角色中的 s3: GetObject/acm-pca:GetCertificateAuthorityCertificate 权限不足。

诊断和解决 RABBITMQ_INVALID_ARN_SSL

要诊断和解决 RABBITMQ_INVALID_ARN_SSL 操作所需的代码，您必须使用亚马逊日志和控制台。CloudWatch

解决无效的 SSL ARN 问题

1. 导航至 Amazon CloudWatch Logs Insights，然后对您的经纪人的日志组运行以下查询`/aws/amazonmq/broker/<broker-id>/general`：

```
fields @timestamp, @message
| sort @timestamp desc
| filter @message like /error.*aws_arn_config/
| limit 10000
```

2. 查找类似于以下内容的错误消息：

```
[error] <0.209.0> aws_arn_config: {<<"could not resolve ARN 'arn:aws:acm-pca:xxxx'
 for configuration 'aws.arns.ssl_options.cacertfile', error: \"AWS service is
 unavailable\"">>,{error,"AWS service is unavailable"}}
```

3. 检查 S3/ACM-PCA 对象并修复所有问题，例如：
   - 验证密钥是否与经纪人位于同一 AWS 区域
   - 确认 ARN 语法正确
   - 确保 IAM 角色具有 s3: GetObject/acm-pca:GetCertificateAuthorityCertificate 权限
4. 更新代理配置并重启代理。

亚马逊 MQ 上的 RabbitMQ：ARN 无效

当代理中配置的一个或多个 ARN 无效或无法访问时，亚马逊 MQ 上的 RabbitMQ 将引发一个 INVALID_ARN 关键操作所需的代码。这适用于用于 SSL 证书、 AWS Secrets Manager 机密、Amazon S3 对象或其他未被更具体的隔离代码（例如 RABBITMQ_INVALID_ARN_LDAP 或 RABBITMQ_INVALID_ASSUMEROLE）涵盖的 AWS 资源引用的 ARN。

在 RABBITMQ_INVALID_ARN 隔离区中的代理可能会遇到功能降级的情况，具体取决于哪些 ARN 无效。依赖于无法访问的资源的功能将不可用，并且代理将记录错误，指出哪个 ARN 无法解析。对代理可用性的影响取决于关键代理操作是否需要无效的 ARN。

诊断和解决 RABBITMQ_INVALID_ARN

要诊断和解决 RABBITMQ_INVALID_ARN 操作所需的代码，您必须使用亚马逊 CloudWatch 日志和受影响资源的相应服务控制台。 AWS

解决无效的 ARN 问题

1. 导航至 Amazon CloudWatch Logs Insights，然后对您的经纪人的日志组运行以下查询`/aws/amazonmq/broker/<broker-id>/general`：

```
fields @timestamp, @message
| sort @timestamp desc
| filter @message like /error.*aws_arn_config/
| limit 10000
```

2. 查找类似于以下内容的错误消息：

```
[error] <0.254.0> aws_arn_config: {<<"could not resolve ARN
 'arn:aws:s3:::bucket-name/certificate.pem' for configuration
 'aws.arns.auth_ldap.ssl_options.cacertfile', error: \"AWS service is unavailable
\"">>,{error,"AWS service is unavailable"}}
```

3. 检查 AWS 资源并修复所有问题，例如：
   - 验证资源是否与代理位于同一 AWS 区域
   - 确认 ARN 语法正确
   - 确保 IAM 角色具有访问资源的相应权限
4. 更新代理配置并重启代理。

适用于 RabbitMQ 的亚马逊 MQ：经纪人无法升级到版本 4

当您尝试将 RabbitMQ 3 代理升级到 RabbitMQ 4 并且该代理具有经典队列或启用 Khepri 元数据存储功能标志时，适用于 RabbitMQ 的 Amazon MQ 将引发需要`RABBITMQ_BROKER_NOT_UPGRADEABLE_TO_V4`操作的代码。Amazon MQ 不会应用主要版本升级，而是会让代理可供发布和使用。

此操作必需代码仅适用于 RabbitMQ 3 代理。要解决此状态并继续升级，请完成以下步骤。

诊断并解决 RABBITMQ_BROKER_NOT_UPGRADEABLE_TO_V4

1. 使用 Amazon MQ 队列迁移工具将所有经典队列迁移到法定队列。该工具可通过 RabbitMQ Web 控制台（“管理员” > “队列迁移”）或 HTTP API 进行访问。
2. 如果在代理上启用了 Khepri，则没有到 RabbitMQ 4 的就地升级路径。改为考虑 RabbitMQ 的蓝绿色部署。

在您解决了潜在问题后，Amazon MQ 会自动清除该状态。`CRITICAL_ACTION_REQUIRED`

> **Note**
>
> 您可以使用 UpdateBrokerAPI 操作将代理引擎版本更新回 3.13 来清除`CRITICAL_ACTION_REQUIRED`状态。

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 Reference》	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。
《Amazon MQ in the AWS CLI Command Reference》	您用来与消息代理结合使用的 AWS CLI 命令的描述。
《Amazon MQ in the AWS CloudFormation User Guide》	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。
区域和端点	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 Getting Started Guide》	RabbitMQ 的官方文档。
《RabbitMQ Client Libraries and Developer Tools》	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 Best Practices》	CloudAMQP 关于使用 RabbitMQ 的最佳实践和建议指南。

Amazon MQ 发行说明

下表列出了 Amazon MQ 特征发布和改进。

日期	文档更新
2026年5月5日	亚马逊 MQ 现在支持从 RabbitMQ 3.13 到 RabbitMQ 4.2 的主要版本就地升级。您的代理端点保持不变，因此无需更改应用程序代码。升级期间，Amazon MQ 会阻止所有连接以执行安全升级，从而导致代理停机。 有关更多信息，请参阅 • 从 RabbitMQ 3.x 升级到 4.x
2026年4月30日	适用于 RabbitMQ 的 Amazon MQ 现在支持运行 RabbitMQ 4.2 及更高版本的经纪商的 Prometheus 指标，允许您将代理可观察性集成到现有的基于 Prometheus 的监控基础设施中。支持以下端点：`/metrics` `/metrics/detailed` 、和`/metrics/memory-breakdown` 。 有关更多信息，请参阅访问 Prometheus 指标和适用于 RabbitMQ 经纪商 CloudWatch 的亚马逊 MQ 可用指标。
2026 年 2 月 19 日	亚马逊 MQ 亚马逊 MQ 现在支持 ActiveMQ 5.19，这是一个新的次要引擎版本。 有关更多信息，请参阅 • ActiveMQ 5.19 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
2026 年 1 月 22 日	亚马逊 MQ 现在支持 RabbitMQ 4.2 及更高版本的经纪商的 JMS 话题交换插件。你可以使用官方 RabbitMQ JMS 客户端在 Amazon MQ for RabbitMQ 代理上运行 JMS 工作负载。它支持 JMS 1.1、2.0 和 3.1。 有关更多信息，请参阅 • JMS 2.0 官方规范（向后兼容 JMS 1.1 并对其进行了扩展）

日期	文档更新
	• JMS 3.1 官方规范 • RabbitMQ JMS 客户端的局限性 • 将你的 JMS 应用程序连接到适用于 RabbitMQ 代理的亚马逊 MQ
2026年1月8日	亚马逊 MQ 现在支持使用 X.509 客户端证书和双向 TLS (mTLS) 配置对 RabbitMQ 4.2 及更高版本的经纪商进行 SSL 证书身份验证。您可以通过 AWS 管理控制台、、 AWS CloudFormation或 AWS CDK 在所有可用 Amazon MQ AWS 区域 的地方配置 SSL 证书身份验证和 mTLS。 AWS CLI 有关更多信息，请参阅 SSL 证书身份验证和配置 mTLS。
2026 年 1 月 6 日	亚马逊 MQ 现在支持 RabbitMQ 4.2 及更高版本上使用外部 HTTP 服务器的代理进行 HTTP 身份验证和授权。您可以通过 AWS 管理控制台、 AWS CloudFormation AWS CLI、或 AWS CDK 在所有可用 Amazon MQ AWS 区域的地方配置 HTTP 身份验证。 有关更多信息，请参阅 HTTP 身份验证和授权。
2025 年 11 月 20 日	亚马逊 MQ 现在支持 RabbitMQ 4.2，这是一个新的主要版本，它引入了对 AMQP 1.0 协议的原生支持、基于 Raft 的新元数据存储 Khepri、本地铲子和法定队列的消息优先级。RabbitMQ 4.2 还包括各种错误修复以及吞吐量和内存管理方面的性能改进。虽然此版本引入了新功能，但仍有一些重大更改。 有关更多信息，请参阅 • RabbitMQ 4 • 开源 RabbitMQ 发行说明 • 配置资源限制 • 支持的协议 • 亚马逊 MQ 版本升级
2025 年 11 月 18 日	亚马逊 MQ 现在支持非洲（开普敦）的 RabbitMQ 支持 Graviton3 的 m7g 实例，大小从中型到 16xlarge 不等。 有关更多信息，请参阅 Amazon MQ for RabbitMQ 代理实例类型。

日期	文档更新
2025 年 11 月 17 日	Amazon MQ 现在支持使用外部 LDAP 目录服务对 RabbitMQ 代理进行 LDAP 身份验证和授权。您可以通过 AWS 管理控制台、 AWS CloudFormation AWS CLI、或 AWS CDK 在所有可用 Amazon MQ AWS 区域 的地方配置 LDAP。 有关更多信息，请参阅 适用于 RabbitMQ 的 Amazon MQ 的 LDAP 身份验证和授权。
2025 年 10 月 22 日	Amazon MQ 现已在亚太地区（新西兰）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。
2025 年 9 月 3 日	Amazon MQ 现在支持使用公共身份提供商 () 对 RabbitMQ 代理进行 OAuth 2.0 身份验证和授权。IdPs您可以通过 AWS 管理控制台、 AWS CloudFormation AWS CLI、或 AWS CDK 在所有可用 Amazon MQ AWS 区域 的地方配置 OAuth 2.0。 有关更多信息，请参阅 OAuth 适用于 RabbitMQ 的亚马逊 MQ 的 2.0 身份验证和授权。
2025 年 7 月 22 日	Amazon MQ 现在支持 Graviton3 驱动的 m7g 实例，用于 RabbitMQ，尺寸范围从中等到 16xlarge。在 m7g 实例上运行的 RabbitMQ 集群，与在 m5 实例上运行的类似 Amazon MQ for RabbitMQ 集群相比，工作负载容量提高高达 50%，吞吐量提升高达 85%。 M7g 实例还具有优化的磁盘卷大小，因实例尺寸而异。有关更多信息，请参阅 Broker instance types。 Amazon MQ 上的 M7g 实例现已在所有一般可用区域推出，除了非洲（开普敦）、加拿大西部（卡尔加里）和欧洲地区（米兰）区域。
2025 年 7 月 8 日	Amazon MQ 现已在亚太地区（台北）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。
2025/4/22	您现在可以使用 `DeleteConfiguration` API 删除 Amazon MQ 代理配置。更多信息，请参阅 Amazon MQ API 参考中的配置。

日期	文档更新
2025 年 4 月 16 日	适用于 RabbitMQ 的 Amazon MQ 现在支持使用双堆栈（IPv4 和 IPv6）终端节点来连接公共和私人经纪人。有关更多信息，请参阅Connecting to Amazon MQ和Configuring a private Amazon MQ broker。
2025 年 4 月 7 日	Amazon MQ 现已在亚太地区（泰国）和墨西哥（中部）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。
2025 年 2 月 13 日	Amazon MQ API FIPS 端点现已在加拿大（中部）和加拿大西部（卡尔加里）区域推出。 有关与 Amazon MQ API 使用 FIPS 端点的更多信息，请参阅 Connecting to Amazon MQ。 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。
2025 年 2 月 12 日	Amazon MQ 宣布以下实例类型的支持终止日期： Broker instance types • ActiveMQ `mq.t2.micro` ：2025 年 5 月 12 日 • ActiveMQ `mq.m4.large` ：2025 年 5 月 12 日 2025 年 3 月 17 日之后，您无法在 `mq.t2.micro` 或 `mq.m4.large` 上创建代理。
2024 年 12 月 10 日	Amazon MQ 现在支持使用 AWS PrivateLink 在您的虚拟私有云 (VPCs) 和 Amazon MQ API 之间进行连接，而无需将您的流量暴露给公共互联网。有关更多信息，请参阅 the section called “使用 AWS PrivateLink 连接到 Amazon MQ”。
2024 年 11 月 18 日	Amazon MQ 现已在亚太地区（马来西亚）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。

日期	文档更新
2024 年 11 月 14 日	Amazon MQ 宣布以下引擎版本的支持终止日期： 管理 Amazon MQ for ActiveMQ 引擎版本 • ActiveMQ 5.17：2025 年 6 月 16 日 管理 Amazon MQ for RabbitMQ 引擎版本 • RabbitMQ 3.11：2025 年 2 月 17 日 • RabbitMQ 3.12：2025 年 3 月 17 日 有关升级到最新版本的更多信息，请参阅 升级 Amazon MQ 代理引擎版本
2024 年 11 月 13 日	Amazon MQ 现在支持双栈服务终端节点，您可以使用或连接这些终端节点。IPv4 IPv6Amazon MQ 双栈区域服务端点可以通过 A 和 AAAA DNS 记录进行解析。有关更多信息，请参阅 ???。
2024 年 7 月 25 日	Amazon MQ 现在支持 ActiveMQ 5.18，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 5.18 发行页面 • 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
2024 年 7 月 22 日	Amazon MQ 现在仅在使用 3.13 及更高版本的代理上支持仲裁队列。仲裁队列是一种复制的 FIFO 队列类型，使用 Raft 共识算法来保持数据一致性。仲裁队列提供毒丸消息处理功能，有助于您管理未处理的消息。 若要开始使用仲裁队列，请参阅 RabbitMQ on Amazon MQ 的仲裁队列。

日期	文档更新
2024 年 7 月 2 日	Amazon MQ for RabbitMQ 现在支持次要版本 RabbitMQ 3.13。对于所有使用引擎版本 3.13 及更高版本的代理，Amazon MQ 会在维护时段内管理升级到所支持的最新补丁版本。有关更多信息，请参阅 升级 Amazon MQ 代理引擎版本。 Amazon MQ for RabbitMQ 大小调整指南 已更新，为使用引擎版本 3.13 的代理加入了队列、每个通道的使用者和 Shovel 的新限制。 有关此版本中的修复和功能的更多信息，请参阅 RabbitMQ 服务器存储库上的 RabbitMQ 3.13 发行说明。 GitHub 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 6 月 10 日	Amazon MQ 现已在加拿大西部（卡尔加里）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。
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。当某个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。Amazon MQ 至少会在引擎版本终止支持前 90 天发出通知。 要查看版本支持日历和支持终止时间，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。

日期	文档更新
2024 年 5 月 9 日	Amazon MQ for RabbitMQ 现在支持次要版本 RabbitMQ 3.12。3.12.13 及更高版本上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。 我们建议代理使用 3.12.13 之前的版本启用队列 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 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 3 月 4 日	Amazon MQ for RabbitMQ 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	Amazon MQ for RabbitMQ 不支持用户名“guest”，并会在您创建新代理时删除默认的访客账户。Amazon MQ 还将定期删除任何由客户创建的名为“guest”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。

日期	文档更新
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。
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。

日期	文档更新
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
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 发行页面 • 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

日期	文档更新
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。
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。

日期	文档更新
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 3.10.10 版本创建新的代理程序。我们建议激活版本升级选项，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。
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。

日期	文档更新
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。
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： • 关于 RabbitMQ 服务器存储库的 RabbitMQ 3.9.24 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 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。

日期	文档更新
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。现在，您可以在 RabbitMQ 队列上启用经典队列版本 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 发行说明 • 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 发行页面 • 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
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 发行页面 • 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
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 发行页面 • 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

日期	文档更新
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。 • 问题排查 • 《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 发行页面 • 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 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。

日期	文档更新
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 在 RabbitMQ 服务器存储库中的发行说明 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 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 “RabbitMQ 的指标”
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。

日期	文档更新
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 “Cross-service 混乱的副手预防”
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 “ActiveMQ 的指标”
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 发行页面 • 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

日期	文档更新
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。
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： • 关于 RabbitMQ 服务器存储库的 RabbitMQ 3.8.17 发行说明 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 管理控制台、 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。 • 计划 Amazon MQ 代理的维护时段

日期	文档更新
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 发行页面 • 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
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。
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，请参阅以下内容：https://docs.aws.amazon.com//amazon-mq/latest/developer-guide/rabbitmq-defaults.html
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 发行页面 • 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”

日期	文档更新
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。
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 发行说明 • 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。

日期	文档更新
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，请参阅入门：创建并连接 RabbitMQ 代理。
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 发行说明 • 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。

日期	文档更新
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 发行说明 • 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。
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 and logging Amazon MQ brokers。
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 发行说明 • 管理 Amazon MQ for ActiveMQ 引擎版本 • 使用 Spring XML 配置文件

日期	文档更新
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。
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 • Monitoring and logging Amazon MQ brokers 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 Monitoring and logging Amazon MQ brokers

日期	文档更新
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
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。 • 选择现有客户托管 CMK — 客户托管 CMKs 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。

日期	文档更新
2019 年 6 月 3 日	有两个新的 Amazon CloudWatch 指标可用：`EstablishedConnectionsCount` 和`InactiveDurableSubscribers` 。有关更多信息，请参阅下列内容： • Monitoring and logging Amazon MQ brokers • Monitoring and logging Amazon MQ brokers
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 • Resource-level 亚马逊 MQ API 操作的权限
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 发行说明 • 管理 Amazon MQ for ActiveMQ 引擎版本 • 使用 Spring XML 配置文件

日期	文档更新
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
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（俄亥俄州） • 美国东部（弗吉尼亚州北部） • 美国西部（北加利福尼亚） • 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） • 欧洲地区（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。

日期	文档更新
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 • networkConnectionStart##

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 发行说明
	  • ActiveMQ 5.15.7 发行说明

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 Users。 • 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。

日期	文档更新

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：
	• 入门：创建并连接 ActiveMQ 代理
	• ActiveMQ 文档中已解决的错误和改进：
	  • ActiveMQ 5.15.6 发行说明
	  • ActiveMQ 5.15.5 发行说明
	  • ActiveMQ 5.15.4 发行说明
	  • ActiveMQ 5.15.3 发行说明
	  • ActiveMQ 5.15.2 发行说明
	  • ActiveMQ 5.15.1 发行说明
	• ActiveMQ 客户端 5.15.6

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Monitoring and logging Amazon MQ brokers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。

日期	文档更新
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（北加利福尼亚） • 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） • 欧洲地区（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 Monitoring and logging Amazon MQ brokers。
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（俄亥俄州） • 美国东部（弗吉尼亚州北部） • 美国西部（北加利福尼亚） • 美国西部（俄勒冈州） • 亚太地区（悉尼） • 欧洲地区（法兰克福） • 欧洲地区（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。

日期	文档更新
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（俄亥俄州） • 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）

日期	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 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 Users。
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。

日期	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。 • > 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Connecting a Java application to your broker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。

日期	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）
	  • 欧洲地区（爱尔兰）
	  使用 mq.t2.micro 实例类型受 CPU 积分和基准性能约束，具备在基准水平之上突增的能力（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。
	  使用 mq.t2.micro 实例类型受 CPU 积分和基准性能约束，具备在基准水平之上突增的能力（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 以编程方式创建和管理经纪人，以及。 AWS SDKs
	• 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
	• 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。

日期	文档更新
	• 您的代理可以发送和接收任何大小的消息。

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。