aws

開發人員指南

# Amazon MQ

# Amazon MQ: 開發人員指南

# Table of Contents

# 什麼是 Amazon MQ？

Amazon MQ 是一種受管訊息代理程式服務，可讓您輕鬆地遷移至雲端中的訊息代理程式。訊息代理程式允許軟體應用程式和元件使用各種程式設計語言、作業系統和正式傳訊通訊協定來進行通訊。目前，Amazon MQ 支持阿帕奇 ActiveMQ 經典和 Rabbit MQ 引擎類型。

Amazon MQ 會與您現有的應用程式和服務搭配運作，而不需要管理、操作或維護自己的傳訊系統。

主題



## Amazon MQ 與 Amazon SQS 或 Amazon SNS 有何不同？

Amazon MQ 則是受管型訊息代理程式服務，可提供與許多熱門訊息代理程式的相容性。我們建議使用 Amazon MQ 從現有訊息代理程式遷移應用程式，這些代理程式需要仰賴與 JMS 或協定 (例如 AMQP 0-9-1、AMQP 1.0、MQTT 和 STOMP) 相容性的現有訊息代理程式。 OpenWire

Amazon SQS 和 Amazon SNS 是可高度擴展、易於使用，而且不需要您設定訊息代理程式的佇列和主題服務。我們會向新應用程式推薦這些服務，因為可受益於幾乎無限制的擴展性和簡單 API。

## 如何開始使用 Amazon MQ？

- 若要使用 Amazon MQ 建立您的第一個代理程式，請參閱 Getting Started with Amazon MQ。
- 若要了解可協助您充分運用 Amazon MQ 的指導方針和注意事項，請參閱 Working with Amazon MQ for ActiveMQ 和 Working with Amazon MQ for RabbitMQ。
- 若要了解 Amazon MQ REST API，請參閱 Amazon MQ REST API 參考。
- 若要進一步了解 Amazon MQ 命 AWS CLI 令，請參閱AWS CLI 命令參考中的 Amazon MQ。

## 我們希望傾聽您的意見

我們誠摯歡迎您提供意見回饋。若要與我們聯絡，請造訪 Amazon MQ 開發論壇。

# 設定 Amazon MQ

您必須先完成下列步驟，才能使用 Amazon MQ。

主題



## 步驟 1：事前準備

### 註冊 AWS 帳戶

如果您還沒有 AWS 帳戶，請完成以下步驟建立新帳戶。

註冊 AWS 帳戶

1. 開啟 https://portal.aws.amazon.com/billing/signup。
2. 請遵循線上指示進行。

   部分註冊程序需接收來電，並在電話鍵盤輸入驗證碼。

   註冊 AWS 帳戶時，會建立 AWS 帳戶根使用者。根使用者有權存取該帳戶中的所有 AWS 服務和資源。作為最佳安全實務，將管理存取權指派給管理使用者，並且僅使用根使用者來執行需要根使用者存取權的任務。

註冊程序完成後，AWS 會傳送一封確認電子郵件給您。您可以隨時登錄 https://aws.amazon.com/ 並選擇 我的帳戶，以檢視您目前的帳戶活動並管理帳戶。

### 建立管理使用者

在您註冊 AWS 帳戶 之後，請保護您的 AWS 帳戶根使用者、啟用 AWS IAM Identity Center，以及建立管理使用者，讓您可以不使用根使用者處理日常作業。

保護您的 AWS 帳戶根使用者

1. 選擇 根使用者 並輸入您的 AWS 帳戶電子郵件地址，以帳戶擁有者身分登入 AWS Management Console。在下一頁中，輸入您的密碼。

   如需使用根使用者登入的說明，請參閱 AWS 登入使用者指南中的以根使用者身分登入。

2. 若要在您的根使用者帳戶上啟用多重要素驗證 (MFA)。

   如需指示，請參閱《IAM 使用者指南》中的為 AWS 帳戶根使用者啟用虛擬 MFA 裝置 (主控台)。

建立管理使用者

1. 啟用 IAM Identity Center。

   如需指示，請參閱AWS IAM Identity Center使用者指南中的啟用 AWS IAM Identity Center。

2. 在 IAM Identity Center 中，將管理存取權授予管理使用者。

   有關如何使用 IAM Identity Center 目錄 作為身分來源的教學課程，請參閱《AWS IAM Identity Center 使用者指南》中的以預設 IAM Identity Center 目錄 設定使用者存取權。

以管理員的身分登入

- 若要使用您的 IAM 身分中心使用者登入，請使用建立 IAM 身分中心使用者時傳送至您電子郵件地址的登入 URL。

  如需有關如何使用 IAM Identity Center 使用者登入的說明，請參閱《AWS 登入 使用者指南》中的登入 AWS存取入口網站。

# 步驟 2：建立使用者並取得您的 AWS 登入資料

若使用者想要與 AWS Management Console 之外的 AWS 互動，則需要程式設計存取權。授予程式設計存取權的方式取決於存取 AWS 的使用者類型。

若要授予使用者程式設計存取權，請選擇下列其中一個選項。

| 哪個使用者需要程式設計存取權？ | 到 | By |
| --- | --- | --- |
| 人力身分<br><br>(IAM Identity Center 中管理的使用者) | 使用臨時憑證簽署對 AWS CLI、AWS SDKs 或 AWS APIs 的程式設計請求。 | 請依照您要使用的介面所提供的指示操作。<br><br>• 關於 AWS CLI，請參閱 AWS Command Line Interface使用者指南 中的 設定 AWS CLI 來使用 AWS IAM Identity Center。<br>• 關於 AWS SDKs、工具和 AWS APIs，請參閱 AWSSDKs 和工具參考指南 中的 IAM Identity Center 驗證。 |
| IAM | 使用臨時憑證簽署對 AWS CLI、AWS SDKs 或 AWS APIs 的程式設計請求。 | 請遵循 IAM 使用者指南 中 使用臨時憑證搭配 AWS 資源 中的指示。 |
| IAM | (不建議使用)<br>使用長期憑證簽署 AWS CLI、AWS SDKs 或 AWS APIs 的程式設計要求。 | 請依照您要使用的介面所提供的指示操作。<br><br>• 關於 AWS CLI，請參閱 AWS Command Line Interface使用者指南 中的 使用 IAM 使用者憑證進行驗證。<br>• 關於 AWS SDKs 和工具，請參閱 AWSSDKs 和工具參考指南 中的 使用長期憑證進行驗證。<br>• 關於 AWS API，請參閱 IAM 使用者指南 中的 管理 IAM 使用者的存取金鑰。 |

# 步驟 3：準備好開始使用範例程式碼

以下教學顯示如何透過 AWS Management Console 使用 Amazon MQ 代理程式，以及如何以程式設計方式連接到 Amazon MQ for ActiveMQ 和 Amazon MQ for RabbitMQ 代理程式。若要使用 ActiveMQ Java 範例程式碼，您必須安裝 Java 標準版開發套件，並對程式碼進行一些變更。

您也可以使用 Amazon MQ REST API 和 AWS SDK，以程式設計方式建立與管理代理程式。

# 後續步驟

既然您已準備好使用 Amazon MQ，請從建立代理程式開始。根據您的代理程式引擎類型，您可以接著將 Java 應用程式連接到 Amazon MQ for ActiveMQ 代理程式，或者使用 RabbitMQ Java 用戶端程式庫將 JVM 型應用程式連接到 Amazon MQ for RabbitMQ 代理程式。

# Amazon MQ 入門

本節將顯示如何建立 Amazon MQ for ActiveMQ 或 RabbitMQ 代理程式，以及如何將您的應用程式連接至該代理程式，協助您更加熟悉 Amazon MQ。

每個代理程式引擎建立和連接至代理程式執行個體的方式稍有不同。選擇下列其中一個您想使用的引擎類型，以取得如何建立和連接至代理程式的詳細資訊。建立並連接至代理程式之後，您也可找到指示來協助您加以刪除。

主題



## 先決條件

開始之前，請完成 Setting Up Amazon MQ 中的步驟。

## 建立並連線至 ActiveMQ 代理程式

代理程式是在 Amazon MQ 上執行的訊息代理程式環境。這是 Amazon MQ 的基本建置區塊。代理程式執行個體類別 (`m5`、`t3`) 和大小 (`large`、`micro`) 的合併說明是代理程式執行個體類型 (例如，`mq.m5.large`)。如需詳細資訊，請參閱 中介裝置。

主題



### 步驟 1：建立 ActiveMQ 代理程式

第一個最常見的 Amazon MQ 任務是建立代理程式。以下範例示範如何使用 AWS Management Console 來建立基本代理程式。

1. 登入 Amazon MQ 主控台。
2. 在 Select broker engine (選取代理程式引擎) 頁面上，選擇 Apache ActiveMQ。
3. 在 Select deployment and storage (選取部署和儲存) 頁面的 Deployment mode and storage type (部署模式和儲存類型) 區段中，執行下列動作：

   a. 選擇 Deployment mode (部署模式) (例如，作用中/待命代理程式)。如需詳細資訊，請參閱 Broker Architecture。

      - 單一執行個體代理程式是由一個可用區域中的一個代理程式組成。代理程式會與您的應用程式以及 Amazon EBS 或 Amazon EFS 儲存磁碟區進行通訊。如需更多詳細資訊，請參閱 Amazon MQ 單一執行個體代理程式。
      - 高可用性的作用中/待命代理程式是由兩個不同可用區域中的兩個代理程式所組成，並設定於備援組合中。這些代理程式會與您的應用程式及 Amazon EFS 同步通訊。如需更多詳細資訊，請參閱 高可用性的 Amazon MQ 作用中/待命代理程式。
      - 如需代理程式網路的範例藍圖詳細資訊，請參閱 藍圖範例。

   b. 選擇 Storage type (儲存體類型) (例如，EBS)。如需更多詳細資訊，請參閱 Storage。

      > **Note**
      >
      > Amazon EBS 會複寫單一可用區域內的資料，且不支援 ActiveMQ 作用中/待命部署模式。

   c. 選擇 Next (下一步)。
4. 在 Configure settings (進行設定) 頁面的 Details (詳細資訊) 區段中，執行以下動作：

   a. 輸入 Broker name (代理程式名稱)。

      > **Important**
      >
      > 請勿在代理程式名稱中加入個人身分識別資訊 (PII) 或其他機密或敏感資訊。包括 CloudWatch Logs 在內的其他 AWS 服務可存取代理程式名稱。代理程式名稱不適用於私有或敏感資料。

   b. 選擇 Broker instance type (代理程式執行個體類型) (例如，mq.m5.large)。如需更多詳細資訊，請參閱 Broker instance types。
5. 在 ActiveMQ Web Console access (ActiveMQ Web 主控台存取) 區段中，提供 Username (使用者名稱) 和 Password (密碼)。以下限制適用於代理程式使用者名稱和密碼：

- 使用者名稱只能包含英數字元、破折號、句點、底線和波狀符號 (- . _ ~)。
- 密碼必須至少有 12 個字元、包含至少 4 個唯一字元，而且不得包含逗號、冒號或等號 (,: =)。

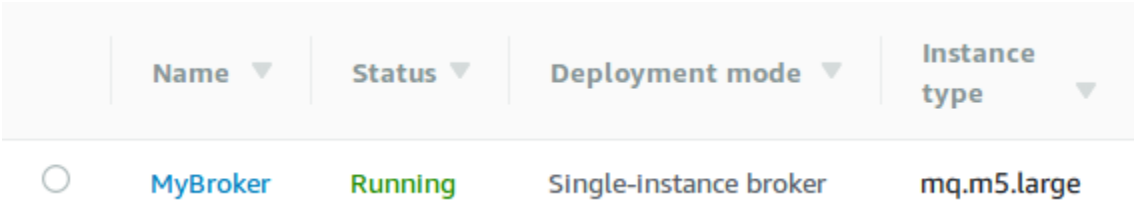
> **Important**
>
> 請勿在代理程式使用者名稱中加入個人身分識別資訊 (PII) 或其他機密或敏感資訊。包括 CloudWatch Logs 在內的其他 AWS 服務可存取代理程式使用者名稱。代理程式使用者名稱不適用於私有或敏感資料。

6. 選擇 部署 。

   當 Amazon MQ 建立您的代理程式時，其會顯示 Creation in progress (正在建立) 狀態。

   建立代理程式大約需要 15 分鐘。

   成功建立代理程式後，Amazon MQ 會顯示 Running (執行中) 狀態。




7. 選擇 ***MyBroker***。

   在 ***MyBroker*** 頁面的 Connect (連線) 區段中，請記下代理程式的 ActiveMQ web console (ActiveMQ Web 主控台) URL，例如：

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

   此外，請記下代理程式的線路通訊協定端點。以下是 OpenWire 端點的範例：

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

# 步驟 2：將 Java 應用程式連接到您的代理程式

建立 Amazon MQ ActiveMQ 代理程式後，您可以將應用程式連接到它。下列範例示範如何使用 Java 訊息服務 (JMS) 建立與代理程式的連線、建立佇列以及傳送訊息。如需完整的作用中 Java 範例，請參閱 Working Java Example。

您可以使用多種 ActiveMQ 用戶端連線至 ActiveMQ 代理程式。建議使用 ActiveMQ 用戶端。

## 先決條件

### 啟用 VPC 屬性

> **Note**
>
> 您無法停用現有 Amazon MQ 代理程式的公開可存取性。

若要確保代理程式可以在 VPC 內存取，您必須啟用 `enableDnsHostnames` 和 `enableDnsSupport` VPC 屬性。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的 VPC 中的 DNS Support。

### 啟用傳入連線

1. 登入 Amazon MQ 主控台。
2. 從代理程式清單中，選擇您的代理程式名稱 (例如，MyBroker)。
3. 在 ***MyBroker*** 頁面的 Connections (連線) 區段中，記下代理程式 Web 主控台 URL 和線路通訊協定的位址和連接埠。
4. 在 Details (詳細資訊) 區段的 Security and network (安全與網路) 下，選擇您的安全群組名稱或 。

   隨即會顯示 EC2 儀表板的 Security Groups (安全群組) 頁面。
5. 從安全群組清單選擇您的安全群組。
6. 在頁面的最下方，選擇 Inbound (傳入)，然後選擇 Edit (編輯)。
7. 在 Edit inbound rules (編輯傳入規則) 對話方塊中，為您要公開存取的每個 URL 或端點新增規則 (下列範例顯示如何針對代理程式 Web 主控台執行此動作)。

   a. 選擇 Add Rule (新增規則)。

   b. 針對 Type (類型)，選擇 Custom TCP (自訂 TCP)。

c. 針對 Port Range (連接埠範圍)，輸入 Web 主控台連接埠 (8162)。

d. 針對 Source (來源)，讓 Custom (自訂) 保持已選取狀態，然後輸入您希望能夠存取 Web 主控台的系統 IP 地址 (例如，`192.0.2.1`)。

e. 選擇 Save (儲存)。

您的代理程式現在已可接受傳入連線。

### 新增 Java 相依性

將 `activemq-client.jar` 和 `activemq-pool.jar` 套件新增到您的 Java 類別路徑。以下範例顯示 Maven 專案 `pom.xml` 檔案中的此等依存關係。

```
<dependencies>
    <dependency>
        <groupId>org.apache.activemq</groupId>
        <artifactId>activemq-client</artifactId>
        <version>5.15.8</version>
    </dependency>
    <dependency>
        <groupId>org.apache.activemq</groupId>
        <artifactId>activemq-pool</artifactId>
        <version>5.15.8</version>
    </dependency>
</dependencies>
```

如需 `activemq-client.jar` 的詳細資訊，請參閱 Apache ActiveMQ 文件中的初始組態。

> **Important**
>
> 在以下的範例程式碼中，生產者和消費者會在單一執行緒中執行。對於生產系統 (或測試代理程式執行個體容錯移轉)，請確定您的生產者和消費者在不同的主機或執行緒上執行。

## 建立訊息生產者和傳送訊息

1. 使用代理程式的端點為訊息生產者建立 JMS 集區連接工廠，然後對工廠呼叫 `createConnection` 方法。

> **Note**
>
> 對於作用中/待命代理程式，Amazon MQ 會提供兩個 ActiveMQ Web 主控台 URL，但一次只有一個作用中的 URL。同樣地，Amazon MQ 為每個線路通訊協定提供兩個端點，但每個配對中一次只有一個作用中的端點。-1 和 -2 尾碼表示備援組合。如需詳細資訊，請參閱 Broker Architecture。
>
> 對於線路通訊協定端點，您可以允許應用程式使用容錯移轉傳輸連線到任一端點。

```
// 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**
>
> 訊息生產者應一律使用 PooledConnectionFactory 類別。如需更多詳細資訊，請參閱一律使用連線集區。

2. 建立工作階段、名為 MyQueue 的佇列和訊息生產者。

```
// 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. 建立訊息字串 "Hello from Amazon MQ!"，然後傳送訊息。

```
// 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. 清除生產者。

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

## 建立訊息消費者和接收訊息

1. 使用代理程式的端點為訊息生產者建立 JMS 連線工廠，然後對工廠呼叫 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**
>
> 訊息消費者不應使用 PooledConnectionFactory 類別。如需更多詳細資訊，請參閱一律使用連線集區。

2. 建立工作階段、名為 MyQueue 佇列和訊息消費者。

```
// 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. 開始等待訊息，並在訊息送達時接收訊息。

```
// 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**
>
> 與 AWS 傳訊服務 (例如 Amazon SQS) 不同，消費者會持續連線到代理程式。

4. 關閉消費者、工作階段和連線。

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

# 步驟 3：(選用) 連線至 AWS Lambda 函數

AWS Lambda 可以連線到 Amazon MQ 代理程式並從中取用訊息。當您將代理程式連接到 Lambda 時，您可以建立事件來源映射，其會讀取佇列中的訊息並同步叫用函數。您建立的事件來源映射會分批讀取來自代理程式的訊息，並以 JSON 物件的形式將它們轉換為 Lambda 承載。

將代理程式連接到 Lambda 函數

1. 將下列 IAM 角色許可新增到 Lambda 函數執行角色。

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

   > **Note**
   >
   > 若沒有必要的 IAM 許可，您的函數將無法成功從 Amazon MQ 資源讀取記錄。

2. (選用) 如果您已建立沒有公開可存取性的代理程式，您必須執行下列其中一項作業，以允許 Lambda 連線到代理程式：

   - 為每個公用子網路設定一個 NAT 閘道。如需詳細資訊，請參閱 AWS Lambda 開發人員指南中的 VPC 連線函數的網際網路和服務存取。
   - 使用 VPC 端點建立 Amazon Virtual Private Cloud (Amazon VPC) 與 Lambda 之間的連線。您的 Amazon VPC 也必須連線至 AWS Security Token Service (AWS STS) 和 Secrets Manager 端點。如需詳細資訊，請參閱 AWS Lambda 開發人員指南中的設定 Lambda 的界面 VPC 端點。

3. 使用 AWS Management Console，針對 Lambda 函數將您的代理程式設定為事件來源。您也可以使用 `create-event-source-mapping` AWS Command Line Interface 命令 。
4. 為 Lambda 函數編寫一些程式碼，以處理從代理程式取用的訊息。事件來源映射擷取的 Lambda 承載取決於代理程式的引擎類型。以下是 Amazon MQ for ActiveMQ 佇列的 Lambda 承載範例。

> **Note**
>
> 在此範例中，`testQueue` 是佇列的名稱。

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

如需有關將 Amazon MQ 連接到 Lambda 的詳細資訊、Lambda 對 Amazon MQ 事件來源支援的選項，以及事件來源映射錯誤的詳細資訊，請參閱 AWS Lambda 開發人員指南中的搭配使用 Lambda 與 Amazon MQ。

## 步驟 4：刪除您的代理程式

如果您沒有使用 Amazon MQ 代理程式（且未預見在不久的將來使用它），最好的做法是從 Amazon MQ 中予以刪除以降低 AWS 成本。

以下範例示範如何使用 AWS Management Console 來刪除代理程式。

1. 登入 Amazon MQ 主控台。
2. 從代理程式清單，選取您的代理程式 (例如 MyBroker)，然後選擇 Delete (刪除)。
3. 在 Delete ***MyBroker***? (刪除 MyBroker?) 對話方塊中，輸入 `delete`，然後選擇 Delete (刪除)。

   刪除代理程式大約需要 5 分鐘。

## 後續步驟

既然您已建立代理程式、已將應用程式連接至其中，並已傳送和接收訊息，那麼您可能想要嘗試以下動作：

- Creating and configuring a broker (其他設定)
- 編輯代理程式引擎版本、執行個體類型、CloudWatch Logs，以及維護偏好設定
- Creating and applying broker configurations
- Listing brokers and viewing broker details
- 建立和管理 ActiveMQ 代理程式使用者
- Rebooting a Broker
- 存取 Amazon MQ 的 CloudWatch 指標

您也可以開始深入探討 Amazon MQ 最佳實務和 Amazon MQ REST API，然後計劃遷移至 Amazon MQ。

# 建立並連線至 RabbitMQ 代理程式

代理程式是在 Amazon MQ 上執行的訊息代理程式環境。這是 Amazon MQ 的基本建置區塊。代理程式執行個體類別 (m5、t3) 和大小 (large、micro) 的合併說明是代理程式執行個體類型 (例如，mq.m5.large)。

主題



## 步驟 1：建立 RabbitMQ 代理程式

第一個最常見的 Amazon MQ 任務是建立代理程式。下列範例顯示如何使用 AWS Management Console 建立基本 Broker。

1. 登入 Amazon MQ 主控台。
2. 在 Select engine (選取引擎) 頁面中，選擇 RabbitMQ，然後選擇 Next (下一步)。
3. 在 Select deployment mode (選取部署模式) 頁面上，選擇 Deployment mode (部署模式)，例如，Cluster deployment (叢集部署)，然後選擇 Next (下一步驟)。

   - 單一執行個體代理程式是由 Network Load Balancer (NLB) 後面的一個可用區域中的一個代理程式組成。代理程式會與您的應用程式以及 Amazon EBS 儲存磁碟區進行通訊。如需詳細資訊，請參閱 單一執行個體代理程式。
   - RabbitMQ cluster deployment for high availability (提供高可用性的 RabbitMQ 叢集部署) 是 Network Load Balancer 後面的三個 RabbitMQ 代理程式節點的邏輯分組，每個節點共用使用者、佇列，以及跨多個可用區域 (AZ) 的分散式狀態。如需詳細資訊，請參閱 高可用性的叢集部署。
4. 在 Configure settings (進行設定) 頁面的 Details (詳細資訊) 區段中，執行以下動作：

   a. 輸入代理程式名稱。

> **Important**
>
> 請勿在代理程式名稱中加入個人身分識別資訊 (PII) 或其他機密或敏感資訊。代理程式名稱可 AWS 供其他服務存取，包括 CloudWatch 記錄檔。代理程式名稱不適用於私有或敏感資料。

b. 選擇代理程式執行個體類型 (例如，mq.m5.large)。如需詳細資訊，請參閱 Broker instance types。

> **Note**
>
> 「其他設定值」段落提供了啟用 CloudWatch 記錄檔和設定代理程式網路存取的選項。如果您在沒有公用存取性的情況下建立私有 RabbitMQ 代理程式，則必須選取 Virtual Private Cloud (VPC) 並設定安全群組來存取代理程式。

5. 在 Configure settings (進行設定) 頁面的 RabbitMQ access (RabbitmQ 存取) 區段上，提供 Username (使用者名稱) 和 Password (密碼)。以下限制適用於代理程式登入認證：

   - 使用者名稱只能包含英數字元、破折號、句點和底線 (- . _)。此值不得包含任何波狀符號 (~) 字元。Amazon MQ 禁止使用 `guest` 作為使用者名稱。
   - 密碼必須至少有 12 個字元、包含至少 4 個唯一字元，而且不得包含逗號、冒號或等號 (,: =)。

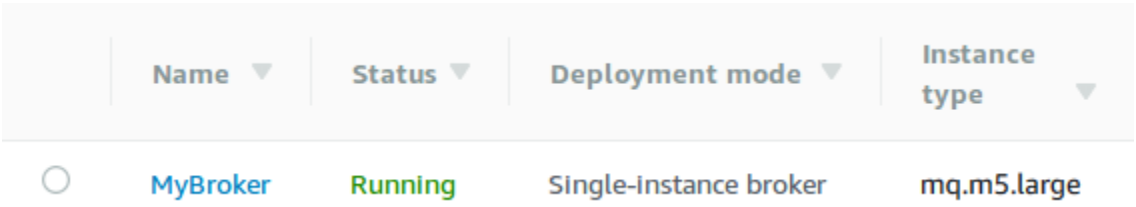
> **Important**
>
> 請勿在代理程式使用者名稱中加入個人身分識別資訊 (PII) 或其他機密或敏感資訊。代理程式使用者名稱可 AWS 供其他服務存取，包括 CloudWatch 記錄檔。代理程式使用者名稱不適用於私有或敏感資料。

6. 選擇下一步。
7. 在 Review and create (檢閱和建立) 頁面上，您可以檢閱您的選取項目，然後視需要編輯它們。
8. 選擇 Create broker (建立代理程式)。

   當 Amazon MQ 建立您的代理程式時，其會顯示 Creation in progress (正在建立) 狀態。

   建立代理程式大約需要 15 分鐘。

成功建立代理程式後，Amazon MQ 會顯示 Running (執行中) 狀態。




9. 選擇***MyBroker***。

   在***MyBroker***頁面的「Connect」區段中，記下代理程式的 RabbitMQ Web 主控台 URL，例如：

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

   此外，請記下代理程式的安全 AMQP 端點。以下是的 amqps 端點公開接聽程式連接埠 5671 的範例。

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

## 步驟 2：將 JVM 型應用程式連接到您的代理程式

建立 RabbitMQ 代理程式後，您可以將應用程式連接到它。下列範例示範如何使用 RabbitMQ Java 用戶端程式庫建立與代理程式的連線、建立佇列以及傳送訊息。您可使用各種語言支援的 RabbitMQ 用戶端程式庫來連線到 RabbitMQ 代理程式。如需支援的 RabbitMQ 用戶端程式庫的詳細資訊，請參閱 RabbitMQ 用戶端程式庫和開發人員工具。

### 必要條件

> **Note**
>
> 下列必要步驟僅適用於在沒有公用存取性的情況下建立的 RabbitMQ 代理程式。如果您正在建立具有公用存取性的代理程式，則可跳過這些步驟。

#### 啟用 VPC 屬性

若要確保代理程式可以在 VPC 內存取，您必須啟用 `enableDnsHostnames` 和 `enableDnsSupport` VPC 屬性。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的 VPC 中的 DNS Support。

啟用傳入連線

1. 登入 Amazon MQ 主控台。
2. 從經紀人列表中，選擇經紀人的名稱（例如，MyBroker）。
3. 在***MyBroker***頁面的「連線」區段中，記下代理程式 Web 主控台 URL 和線路層級通訊協定的位址和連接埠。
4. 在 Details (詳細資訊) 區段的 Security and network (安全與網路) 下，選擇您的安全群組名稱或 。

   隨即會顯示 EC2 儀表板的 Security Groups (安全群組) 頁面。
5. 從安全群組清單選擇您的安全群組。
6. 在頁面的最下方，選擇 Inbound (傳入)，然後選擇 Edit (編輯)。
7. 在 Edit inbound rules (編輯傳入規則) 對話方塊中，為您要公開存取的每個 URL 或端點新增規則 (下列範例顯示如何針對代理程式 Web 主控台執行此動作)。

   a. 選擇 Add Rule (新增規則)。

   b. 針對 Type (類型)，選擇 Custom TCP (自訂 TCP)。

   c. 針對 Source (來源)，讓 Custom (自訂) 保持已選取狀態，然後輸入您希望能夠存取 Web 主控台的系統 IP 地址 (例如，`192.0.2.1`)。

   d. 選擇 Save (儲存)。

      您的代理程式現在已可接受傳入連線。

新增 Java 相依性

如果您使用 Apache Maven 自動建置，請將以下相依性新增至 `pom.xml` 檔案。如需有關 Apache Maven 中專案物件模型檔案的詳細資訊，請參閱 POM 簡介。

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

如果您使用 Gradle 自動建置，請宣告以下相依性。

```
dependencies {
```

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

### 匯入 **Connection** 和 **Channel** 類別

RabbitMQ Java 用戶端使用 com.rabbitmq.client 作為其頂層套件，而 Connection 和 Channel API 類別分別代表 AMQP 0-9-1 連線和通道。在使用前匯入 Connection 和 Channel 類別，如以下範例所示。

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

### 建立 **ConnectionFactory** 並連接到您的代理程式

使用以下範例，透過給定的參數建立 ConnectionFactory 類別的執行個體。使用 setHost 方法來設定您稍早記下的代理程式端點。對於 AMQPS 線路層級連線, 使用連接埠 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();
```

### 將訊息發佈至交換

您可使用 Channel.basicPublish 將訊息發佈至交換。下列範例使用 AMQP Builder 類別來建置具有內容類型 plain/text 的訊息屬性對象。

```
byte[] messageBodyBytes = "Hello, world!".getBytes();
channel.basicPublish(exchangeName, routingKey,
```

```
            new AMQP.BasicProperties.Builder()
              .contentType("text/plain")
              .userId("userId")
              .build(),
              messageBodyBytes);
```

> **Note**
>
> 請注意，BasicProperties 是自動產生的持有人類別 AMQP 的內部類別。

訂閱佇列並接收訊息

您可以使用 Consumer 界面，藉由訂閱佇列來接收訊息。訂閱後，訊息就會在送達時自動傳送。

實作 Consumer 的最簡單方法是使用子類別 DefaultConsumer。DefaultConsumer 物件可以作為 basicConsume 呼叫的一部分來傳遞，以設定訂閱，如以下範例所示。

```
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**
>
> 因為我們指定了 autoAck = false，則必須認可傳送至 Consumer 的訊息，最方便在 handleDelivery 方法中進行，如範例所示。

關閉您的連線並中斷代理程式的連線

為了中斷與 RabbitMQ 代理程式的連線，請關閉通道和連線，如下面所示。

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

> **Note**
>
> 如需使用 RabbitMQ Java 用戶端程式庫的詳細資訊，請參閱 RabbitMQ Java 用戶端 API 指南。

## 步驟 3：（可選）連接到 AWS Lambda 功能

AWS Lambda 可以連線到 Amazon MQ 代理程式並使用來自您的訊息。當您將代理程式連接到 Lambda 時，您可以建立事件來源映射，其會讀取佇列中的訊息並同步叫用函數。您建立的事件來源映射會分批讀取來自代理程式的訊息，並以 JSON 物件的形式將它們轉換為 Lambda 承載。

將代理程式連接到 Lambda 函數

1. 將下列 IAM 角色許可新增到 Lambda 函數執行角色。

   - mq：DescribeBroker
   - ec2：CreateNetworkInterface
   - ec2：DeleteNetworkInterface
   - ec2：DescribeNetworkInterfaces
   - ec2：DescribeSecurityGroups
   - ec2：DescribeSubnets
   - ec2：DescribeVpcs
   - 日誌：CreateLogGroup
   - 日誌：CreateLogStream
   - 日誌：PutLogEvents
   - 秘書經理:GetSecretValue

> **Note**
>
> 若沒有必要的 IAM 許可，您的函數將無法成功從 Amazon MQ 資源讀取記錄。

2. (選用) 如果您已建立沒有公開可存取性的代理程式，您必須執行下列其中一項作業，以允許 Lambda 連線到代理程式：

   - 為每個公用子網路設定一個 NAT 閘道。如需詳細資訊，請參閱 AWS Lambda 開發人員指南中的 VPC 連線函數的網際網路和服務存取。
   - 使用 VPC 端點建立 Amazon Virtual Private Cloud (Amazon VPC) 與 Lambda 之間的連線。您的 Amazon VPC 人雲端也必須連線到 AWS Security Token Service (AWS STS) 和機 Secrets Manager 端點。如需詳細資訊，請參閱 AWS Lambda 開發人員指南中的設定 Lambda 的界面 VPC 端點。

3. 使用 AWS Management Console，針對 Lambda 函數將您的代理程式設定為事件來源。您也可以使用`create-event-source-mapping` AWS Command Line Interface 指令。

4. 為 Lambda 函數編寫一些程式碼，以處理從代理程式取用的訊息。事件來源映射擷取的 Lambda 承載取決於代理程式的引擎類型。以下是 Amazon MQ for RabbitMQ 佇列的 Lambda 承載範例。

> **Note**
>
> 在範例中，`test` 是佇列的名稱，`/` 是預設虛擬主機的名稱。接收訊息時，事件來源會在 `test::/` 列出訊息。

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

如需有關將 Amazon MQ 連接到 Lambda 的詳細資訊、Lambda 對 Amazon MQ 事件來源支援的選項，以及事件來源映射錯誤的詳細資訊，請參閱 AWS Lambda 開發人員指南中的搭配使用 Lambda 與 Amazon MQ。

## 步驟 4：刪除您的代理程式

如果您不使用 Amazon MQ 代理程式 (並且不預見在不久的 future 使用)，最佳做法是將其從 Amazon MQ 刪除以降低成本。 AWS

以下範例示範如何使用 AWS Management Console來刪除代理程式。

1. 登入 Amazon MQ 主控台。
2. 從經紀人清單中，選取您的經紀人 (例如，MyBroker)，然後選擇 [刪除]。
3. 在刪除 ***MyBroker*** ？」對話方塊中，鍵入，`delete`然後選擇「刪除」。

   刪除代理程式大約需要 5 分鐘。

## 後續步驟

既然您已建立代理程式、已將應用程式連接至其中，並已傳送和接收訊息，那麼您可能想要嘗試以下動作：

- 編輯代理程式引擎版本、執行個體類型、CloudWatch Logs，以及維護偏好設定
- Listing brokers and viewing broker details
- 建立和管理 ActiveMQ 代理程式使用者
- Rebooting a Broker
- 存取 Amazon MQ 的 CloudWatch 指標

您還可開始深入探討 Amazon MQ 最佳實務和 Amazon MQ REST API，然後計劃遷移至 Amazon MQ。

# 管理 Amazon MQ 代理程式

在下列各節中，您可以找到管理和維護 Amazon MQ 代理程式的指示。

主題



# 維護 Amazon MQ 代理程式

Amazon MQ 會定期對訊息代理程式的硬體、作業系統或引擎軟體執行維護。維護的持續時間會有所不同，但最多可能持續兩小時，視您針對訊息代理程式排程的操作而定。例如，如果您已啟用自動次要引擎版本升級，或變更代理程式執行個體類型，Amazon MQ 將在下一個排定的維護時段套用變更。

若要將維護時段的停機時間降到最低，建議您選取跨多個可用區域 (AZ) 且具有高可用性的代理程式部署模式。根據您的代理程式引擎類型，Amazon MQ 提供以下異地同步備份部署模式。

- Amazon MQ for ActiveMQ — Amazon MQ for ActiveMQ 提供作用中/待命部署，以獲得高可用性。在作用中/待命模式中，Amazon MQ 一次執行一個執行個體的維護操作，以確保至少有一個執行個體保持可用狀態。此外，您可以設定代理程式網路且維護時段分散在整個星期。
- Amazon MQ for RabbitMQ – Amazon MQ for RabbitMQ 提供叢集部署，以獲得高可用性。在叢集部署中，Amazon MQ 會執行維護操作 (一次一個節點)，一直保持至少兩個執行中的節點。

如需 Amazon MQ 建議的最佳實務詳細資訊，以確保代理程式在維護時段期間和之後有效執行，請參閱下列文件中您的代理程式引擎類型。

- the section called “Amazon MQ for ActiveMQ 最佳實務”
- the section called “Amazon MQ for RabbitMQ 最佳實踐”

您可以排定每週在指定的時間進行一次維護，最多持續兩小時。這會設定來自 Amazon MQ 的維護動作要排程和啟動的時段。

您可以在第一次建立代理程式時排定維護時段，或經由更新代理程式偏好設定。下列主題說明使用 AWS Management Console、 AWS CLI和 Amazon MQ API 調整代理程式維護時段。

主題

- 調整代理程式維護時段

## 調整代理程式維護時段

在您選取的維護時段期間，Amazon MQ 將執行任何擱置中的變更，例如自動次要版本升級。若要調整代理程式維護時段 AWS Management Console，您可以使用 AWS CLI、或 Amazon MQ API。

> **Important**
>
> 您只能將代理程式的維護時段調整為在下一個排定的維護時段前最多四次。Amazon MQ 會套用四個維護時段調整的限制，以確保重要的軟體和安全修補程式以及重要的硬體升級不會無限期延遲和延期。
>
> 代理程式維護時段完成後，Amazon MQ 會重設限制，讓您在下一個維護時段發生前調整排程。
>
> 調整代理程式維護時段時，不會影響代理程式可用性。

### AWS Management Console

若要調整代理程式維護時段，請使用 AWS Management Console

1. 登入 Amazon MQ 主控台。
2. 在左側導覽窗格中，選擇 Brokers (代理程式)，然後從清單中選擇您要升級的代理程式。
3. 在代理程式詳細資訊頁面上，選擇 Edit (編輯)。
4. 在 Maintenance (維護) 之下，執行下列動作。

    a. 針對 Start day (開始日)，從下拉式清單中選擇星期幾，例如週日。

b. 針對 Start time (開始時間)，選擇您想排定下一個代理程式維護時段的一天中的小時和分鐘，例如 12:00。

> **Note**
>
> Start time (開始時間) 選項設定於 UTC+0 時區。

5. 捲動到頁面底部，然後選擇 Save (儲存)。維護時段會立即調整。
6. 在代理程式詳細資訊頁面的 Maintenance widnow (維護時段) 下，確認已顯示新的偏好排程。

AWS CLI

若要使用調整代理程式維護視窗 AWS CLI

1. 使用 update-broker CLI 命令並指定下列參數，如範例所示。

   - `--broker-id` – Amazon MQ 針對代理程式產生的唯一 ID。您可以從代理程式 ARN 解析 ID。例如，假定是以下 ARN `arn:aws:mq:us-east-2:123456789012:broker:MyBroker:b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9`，代理程式 ID 會是 `b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9`。
   - `--maintenance-window-start-time` – 決定下列結構中提供的每週維護時段開始時間的參數。
     - `DayOfWeek` – 星期幾，使用下列語法：`MONDAY| TUESDAY | WEDNESDAY | THURSDAY | FRIDAY | SATURDAY | SUNDAY`
     - `TimeOfDay` – 24 小時制的時間。
     - `TimeZone` – (選用) 國家/城市或 UTC 位移格式的時區。預設為 UTC 。

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

2. (選用) 使用 describe-broker CLI 命令，以確認已成功更新維護時段。

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

Amazon MQ API

使用 Amazon MQ API 調整代理程式維護時段

1. 使用 UpdateBroker API 操作。指定 broker-id 作為路徑參數。下列範例假設 us-west-2 地區中的代理程式。如需可用 Amazon MQ 端點的詳細資訊，請參閱 AWS 一般參考 中的 Amazon MQ 端點和配額。

```
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
```

在請求承載中使用 maintenanceWindowStartTime 參數和 WeeklyStartTime 資源類型。

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

2. (選擇性) 使用 DescribeBrokerAPI 作業驗證維護時段是否已成功更新。 broker-id指定為路徑參數。

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

# 升級 Amazon MQ 代理程式引擎版本

Amazon MQ 為所有支援的代理程式引擎類型提供新的代理程式引擎版本。新的引擎版本可能包括安全修補程式、錯誤修正和其他代理程式引擎改進。Amazon MQ 支援新的引擎版本時，您可以控制如何及何時升級您的代理程式。

代理程式引擎版本會組織為 X.Y.Z。在每個引擎類型的 Amazon MQ 實作中，X.Y 被視為主要版本，而 Z 被視為次要版本。有兩種類型的升級：

- 主要版本升級 – 發生於主要引擎版本號碼變更時。例如，從版本 1.0 升級至版本 1.1 被視為主要版本升級。
- 次要版本升級 – 僅發生於次要引擎版本號碼變更時。例如，從 1.1.0 升級至版本 1.1.1 被視為次要版本升級。

如需每個特定代理程式引擎類型的主要和次要版本管理的詳細資訊，請參閱下列主題。

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

當您啟用自動次要版本升級選項時，Amazon MQ 會在新的次要版本可用時將代理程式升級為新版本。只有在代理程式執行的次要引擎版本低於新建議的次要版本時，會發生自動次要版本升級。對於主要升級，您必須手動升級引擎版本。

手動和自動版本升級兩著會在排定的維護時段期間或在重新啟動代理程式之後發生。

下列主題說明如何手動升級代理程式引擎版本，以及啟用自動次要版本升級。

主題

- 手動升級引擎版本
- 自動升級次要引擎版本

## 手動升級引擎版本

若要手動將代理程式的引擎版本升級至新的主要或次要版本，您可以使用AWS Management Console、AWS CLI 或 Amazon MQ API。

AWS Management Console

使用 AWS Management Console 升級代理程式的引擎版本

1. 登入 Amazon MQ 主控台。
2. 在左側導覽窗格中，選擇 Brokers (代理程式)，然後從清單中選擇您要升級的代理程式。
3. 在代理程式詳細資訊頁面上，選擇 Edit (編輯)。

4. 在 Specifications (規格) 之下，針對 Broker engine version (代理程式引擎版本)，從下拉式清單中選擇新的版本號碼。
5. 捲動到頁面底部，然後選擇 Schedule modification (排程修改)。
6. 在 Schedule broker modifications (排定代理程式修改) 頁面上，針對 When to apply modifications (套用修改的時機)，選擇下列其中一項。
   - 如果您希望 Amazon MQ 在下一個排定的維護時段完成版本升級，請選擇 After the next reboot (在下次重新啟動後)。
   - 如果您想要重新啟動代理程式並立即升級引擎版本，請選擇 Immediately (立即)。

   > **Important**
   >
   > 您的代理程式會在重新啟動時離線。

7. 選擇 Apply (套用) 以完成變更套用。

AWS CLI

使用 AWS CLI 升級代理程式的引擎版本

1. 使用 update-broker CLI 命令並指定下列參數，如範例所示。
   - `--broker-id` – Amazon MQ 針對代理程式產生的唯一 ID。您可以從代理程式 ARN 解析 ID。例如，假定是以下 ARN `arn:aws:mq:us-east-2:123456789012:broker:MyBroker:b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9`，代理程式 ID 會是 `b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9`。
   - `--engine-version` – 要升級至的代理程式引擎版本號碼。

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

2. (選用) 如果您想立即升級引擎版本，請使用 reboot-broker CLI 命令重新啟動代理程式。

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

   如果您不想立即重新啟動代理程式並套用變更，Amazon MQ 會在下一個排定的維護時段期間升級代理程式。

> **Important**
>
> 您的代理程式會在重新啟動時離線。

### Amazon MQ API

使用 Amazon MQ API 升級代理程式的引擎版本

1. 使用 UpdateBroker API 操作。指定 `broker-id` 作為路徑參數。下列範例假設 `us-west-2` 地區中的代理程式。如需可用 Amazon MQ 端點的詳細資訊，請參閱 AWS 一般參考 中的 Amazon MQ 端點和配額。

   ```
   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
   ```

   在請求承載中使用 `engineVersion`，以指定要升級至代理程式版本號碼。

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

2. (選用) 如果您想立即升級引擎版本，請使用 RebootBroker API 操作重新啟動代理程式。`broker-id` 被指定為路徑參數。

   ```
   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
   ```

   如果您不想立即重新啟動代理程式並套用變更，Amazon MQ 會在下一個排定的維護時段期間升級代理程式。

> **Important**
>
> 您的代理程式會在重新啟動時離線。

## 自動升級次要引擎版本

您可以控制是否在第一次建立代理程式時，或藉由修改代理程式偏好設定，為代理程式啟用自動次要版本升級。若要啟用現有代理程式的自動次要版本升級，您可以使用 AWS Management Console、AWS CLI 或 Amazon MQ API。

AWS Management Console

使用 AWS Management Console 啟用自動次要版本升級

1. 登入 Amazon MQ 主控台。
2. 在左側導覽窗格中，選擇 Brokers (代理程式)，然後從清單中選擇您要升級的代理程式。
3. 在代理程式詳細資訊頁面上，選擇 Edit (編輯)。
4. 在 Maintenance (維護) 之下，選擇 Enable automatic minor version upgrades (啟用自動次要版本升級)。

   > **Note**
   >
   > 如果已選取此選項，您不需要進行任何變更。

5. 選擇頁面底部的 Save (儲存)。

AWS CLI

若要透過 AWS CLI 來啟用自動次要版本升級，請使用 update-broker CLI 命令並指定下列參數。

- `--broker-id` – Amazon MQ 針對代理程式產生的唯一 ID。您可以從代理程式 ARN 解析 ID。例如，假定是以下 ARN `arn:aws:mq:us-east-2:123456789012:broker:MyBroker:b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9`，代理程式 ID 會是 `b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9`。
- `--auto-minor-version-upgrade` – 啟用自動次要版本升級選項。

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

如果您想要為代理程式停用自動次要版本升級，請使用 `--no-auto-minor-version-upgrade` 參數。

Amazon MQ API

若要透過 Amazon MQ API 來啟用自動次要版本升級，請使用 UpdateBroker API 操作。指定 `broker-id` 作為路徑參數。下列範例假設 `us-west-2` 地區中的代理程式。如需可用 Amazon MQ 端點的詳細資訊，請參閱 AWS 一般參考 中的 Amazon MQ 端點和配額。

```
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
```

在請求承載中使用 `autoMinorVersionUpgrade` 屬性，以啟用自動次要版本升級。

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

如果您想要為代理程式停用自動次要版本升級，請在請求承載中設定 `"autoMinorVersionUpgrade": "false"`。

# 代理程式狀態

狀態會指出代理程式的目前狀況。下表列出 Amazon MQ 代理程式的狀態。

| 主控台 | API | 描述 |
| --- | --- | --- |
| 建立失敗 | CREATION_FAILED | 無法建立代理程式。 |
| 正在建立 | CREATION_IN_PROGRESS | 目前正在建立代理程式。 |
| 正在刪除 | DELETION_IN_PROGRESS | 目前正在刪除代理程式。 |
| 正在重新啟動 | REBOOT_IN_PROGRESS | 目前正在重新啟動代理程式。 |

| 主控台 | API | 描述 |
| --- | --- | --- |
| 執行中 | RUNNING | 代理程式可運作。 |
| 需執行的關鍵動作 | CRITICAL_ACTION_REQUIRED | 代理程式正在執行，但處於降級狀態，需要立即採取動作。您可以在 the section called “故障診斷：需執行的 Amazon MQ 動作的代碼” 中選擇需採取動作的代碼，以找到解決問題的說明。 |

# 列出 Amazon MQ 代理程式和檢視代理程式詳細資訊

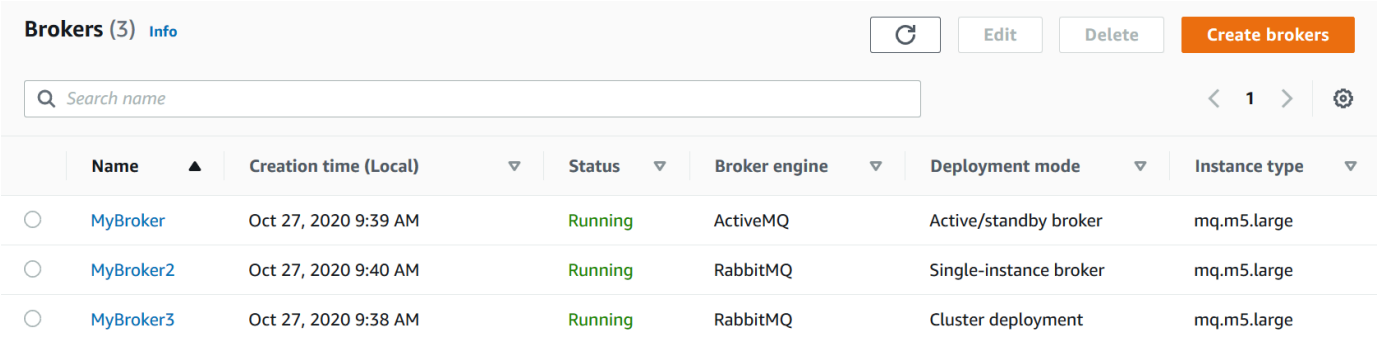
當您請求 Amazon MQ 建立代理程式時，建立程序大約需要 15 分鐘。

以下範例顯示如何使用 AWS Management Console，列出目前區域中您的代理程式，來確認代理程式是否存在。

## 列出代理程式和檢視代理程式詳細資訊

1. 登入 Amazon MQ 主控台。

   列出目前區域中您的代理程式。



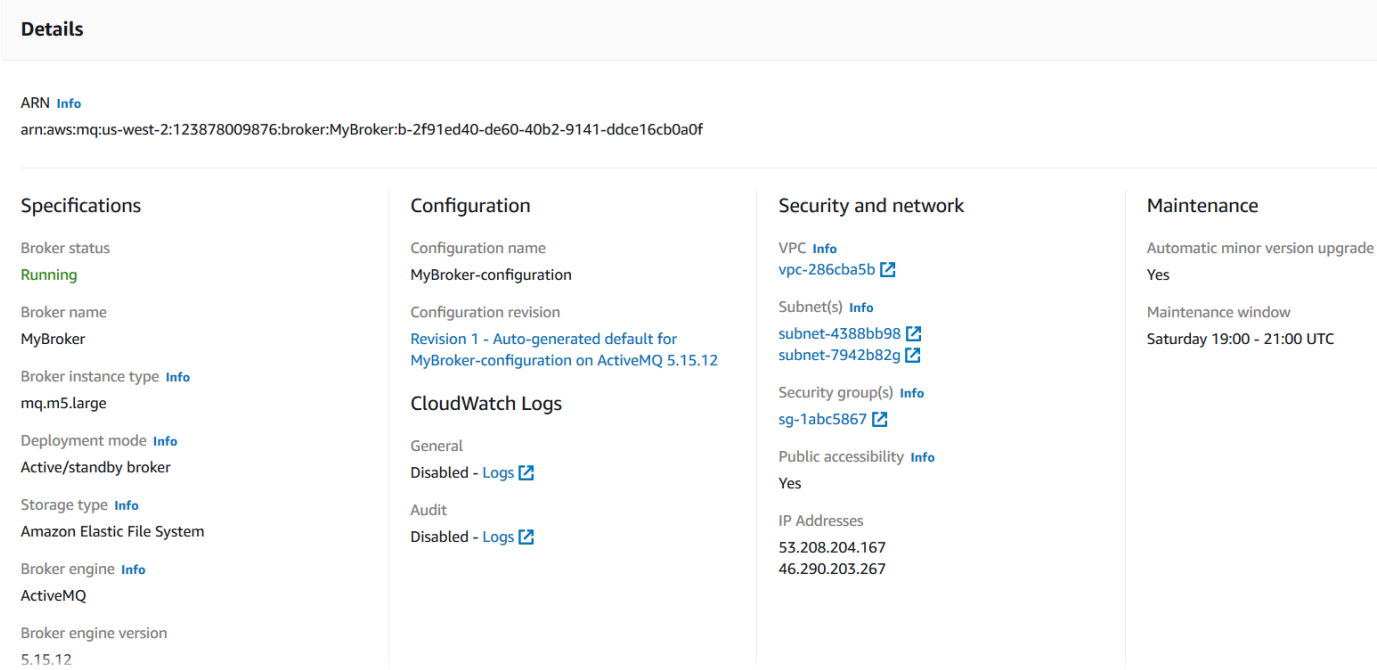

   每個代理程式都會顯示下列資訊：

   - 名稱
   - Creation (建立) 日期
   - 狀態

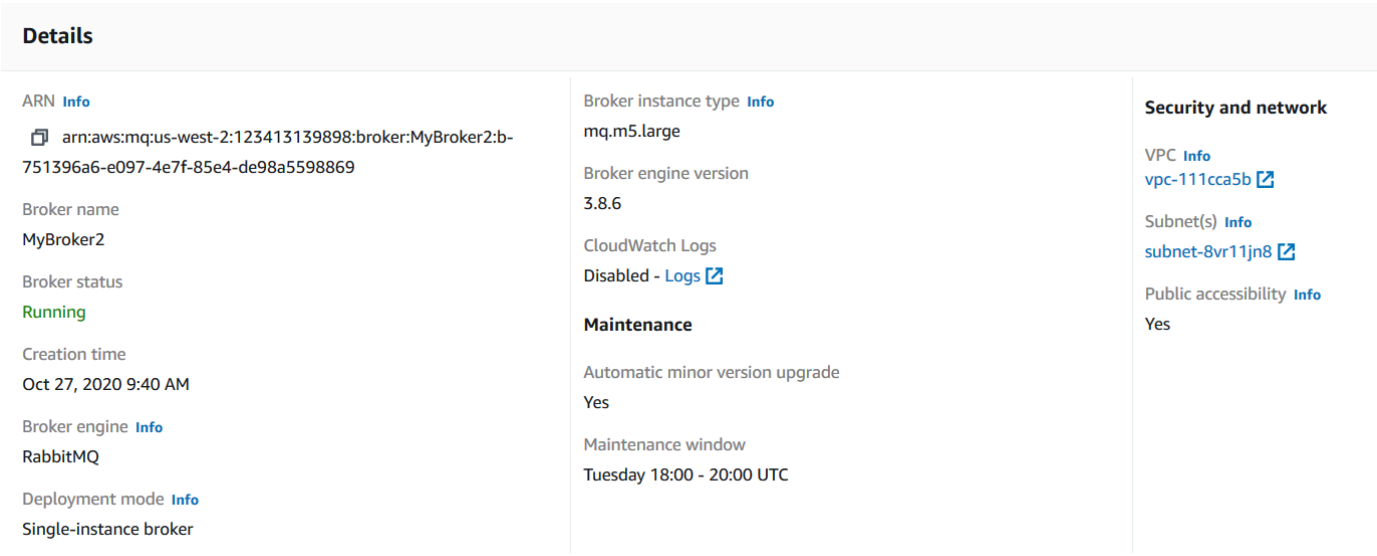
- Deployment mode (部署模式)
- 執行個體類型

2. 選擇您的代理程式名稱。

針對 ActiveMQ 代理程式，在 ***MyBroke*** 頁面上，會顯示已為您的代理程式設定的 Details (詳細資訊)：

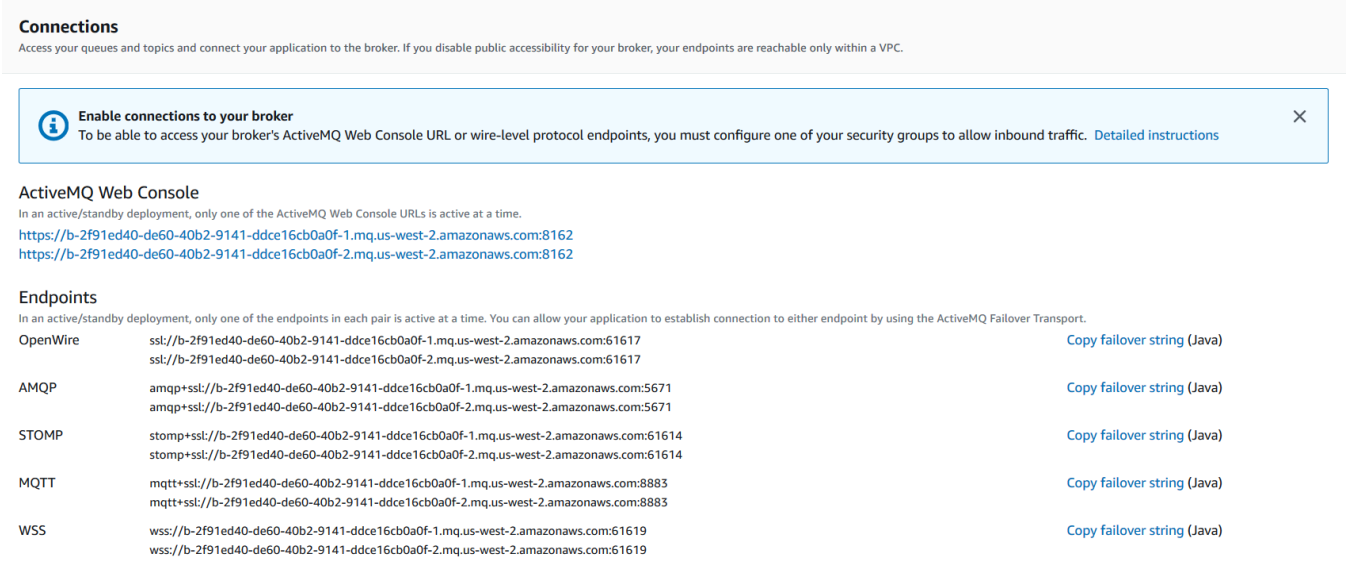



對於 Amazon MQ for RabbitMQ 代理程式，您可以在 ***MyBroker2*** 頁面的 Details (詳細資訊) 區段下檢視所選的設定，如下所示。

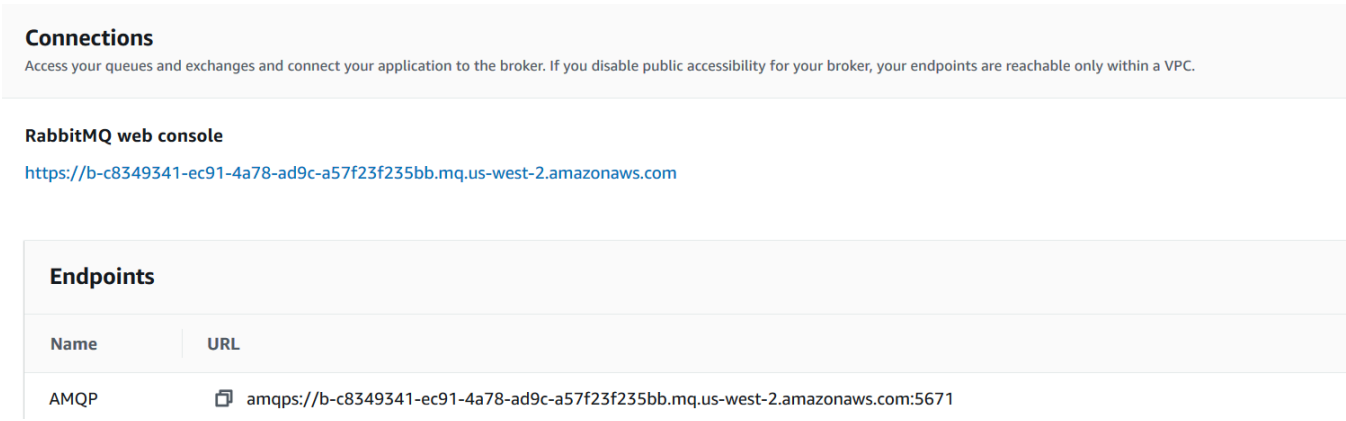


在 Details (詳細資訊) 區段下會顯示以下資訊：

- 在 Connections (連線) 區段中，針對 Amazon MQ for ActiveMQ 代理程式，顯示 Web 主控台 URL 和線路通訊協定端點。




在 Connections (連線) 區段中，針對 Amazon MQ for RabbitMQ 代理程式，顯示 Web 主控台 URL 和安全 AMQP 端點。




- 針對 Amazon MQ for ActiveMQ 代理程式，在 Users (使用者) 區段中，顯示與代理程式相關聯的使用者

> **Important**
>
> Amazon MQ for RabbitMQ 代理程式不支援透過 AWS Management Console 和 Amazon MQ API 管理使用者。

# 在沒有公開存取性的情況下存取代理程式 Web 主控台

如果您停用代理程式的公開存取性，則必須執行下列步驟，才能存取代理程式的 Web 主控台。

> **Note**
>
> VPC 和安全群組的名稱是以下範例特有的。

## 先決條件

若要執行下列步驟，您必須設定下列項目：

- VPC
  - 沒有網際網路閘道且 Amazon MQ 代理程式連接至其中的 VPC，名為 `private-vpc`。
  - 第二個 VPC 具有網際網路閘道，名為 `public-vpc`。
  - 這兩個 VPC 必須連線 (例如，使用 VPC 對等互連)，以便公有 VPC 中的 Amazon EC2 執行個體可與私有 VPC 中的 EC2 執行個體通訊。
  - 如果您使用 VPC 對等互連，則必須將這兩個 VPC 的路由表設定為可供對等連線使用。
- 安全群組
  - 建立 Amazon MQ 代理程式所用的安全群組，名為 `private-sg`。
  - `public-vpc` VPC 中用於 EC2 執行個體的第二個安全群組，名為 `public-sg`。
  - `private-sg` 必須允許來自 `public-sg` 的傳入連線。我們建議將此安全群組限制為 ActiveMQ 的連接埠 8162，以及 RabbitMQ 的連接埠 443。
  - `public-sg` 必須允許來自連接埠 22 上機器的傳入連線。

## 在沒有公開存取性的情況下存取代理程式的 Web 主控台

1. 在 `public-vpc` 中建立 Linux EC2 執行個體 (必要時，具有公有 IP)。
2. 若要驗證是否正確地設定 VPC，請建立與 EC2 執行個體的 `ssh` 連線，然後使用 `curl` 命令與代理程式的 URI 搭配。
3. 從您的機器，使用私有金鑰檔案的路徑，以及私有 EC2 執行個體的 IP 地址來建立 EC2 執行個體的 `ssh` 通道。例如：

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

在您的機器上啟動轉送代理伺服器。

4. 在您的機器上安裝代理用戶端，例如 FoxyProxy。
5. 使用以下設定來設定您的代理用戶端：

   - 針對代理類型，指定 SOCKS5。
   - 對於 IP 地址、DNS 名稱和伺服器名稱，指定 `localhost`。
   - 對於連接埠，指定 8080。
   - 移除任何現有的 URL 模式。
   - 針對 URL 模式，指定 `*.mq.*.amazonaws.com*`
   - 針對連線類型，指定 HTTP(S)。

   啟用您的代理用戶端時，您可以在機器上存取 Web 主控台。

# 重新啟動 Amazon MQ 代理程式

若要將新組態套用至代理程式，您可以重新啟動代理程式。

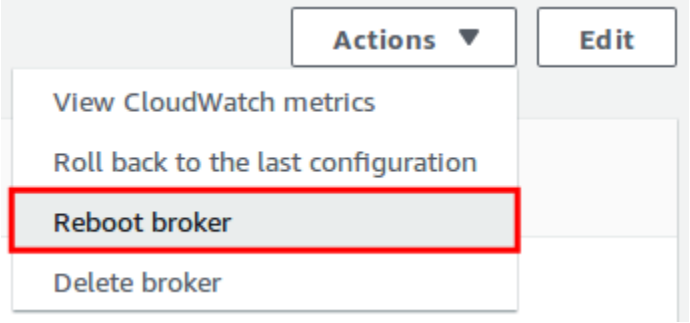
> **Note**
>
> 如果您的 ActiveMQ 代理程式無法回應，您可以將它重新啟動，以從故障狀態復原。

以下範例示範如何使用 AWS Management Console 來重新啟動 Amazon MQ 代理程式。

## 重新啟動 Amazon MQ 代理程式

1. 登入 Amazon MQ 主控台。
2. 從代理程式清單中，選擇您的代理程式名稱 (例如，MyBroker)。
3. 在 ***MyBroker*** 頁面，選擇 Actions (動作)、Reboot broker (重新啟動代理程式)。

   > **Important**
   >
   > 單一執行個體代理程式會在重新啟動時離線。叢集代理程式將可使用，但每個節點會逐一重新啟動。




4. 在 Reboot broker (重新啟動代理程式) 對話方塊中，選擇 Reboot (重新開機)。

   重新啟動代理程式大約需要 5 分鐘。如果重新啟動包含執行個體大小變更，或是在佇列深度較大的代理程式上執行，則重新啟動程序可能需要更長的時間。

# 刪除 Amazon MQ 代理程式

如果您沒有使用 Amazon MQ 代理程式（且未預見在不久的將來使用它），最好的做法是從 Amazon MQ 中予以刪除以降低 AWS 成本。

以下範例示範如何使用 AWS Management Console 來刪除代理程式。

## 刪除 Amazon MQ 代理程式

1. 登入 Amazon MQ 主控台。
2. 從代理程式清單，選取您的代理程式 (例如 MyBroker)，然後選擇 Delete (刪除)。
3. 在 Delete ***MyBroker***? (刪除 MyBroker?) 對話方塊中，輸入 `delete`，然後選擇 Delete (刪除)。

   刪除代理程式大約需要 5 分鐘。

# 管理 Amazon MQ 代理程式組態

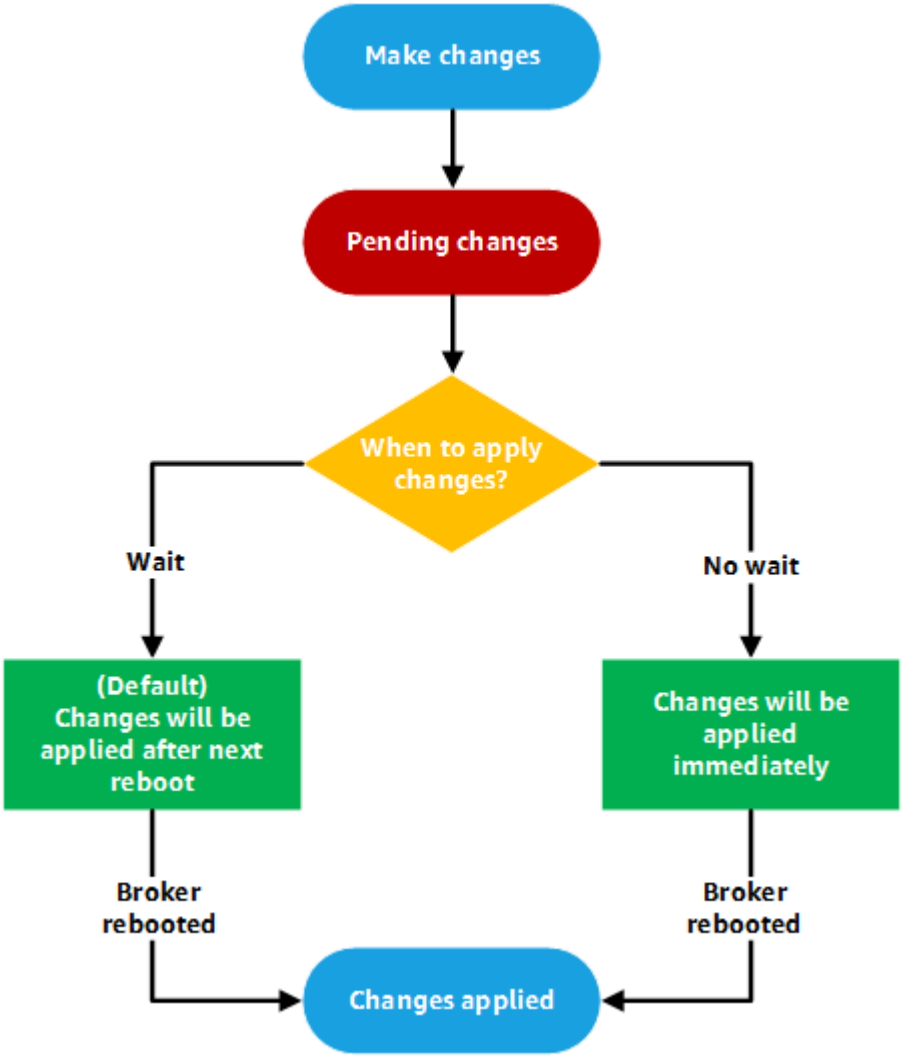
組態包含代理程式的所有設定。您可以在建立任何代理程式之前建立組態。然後，您可以將組態套用至一或多個代理程式。

## Amazon MQ 代理程式組態生命週期

對組態修訂版或 ActiveMQ 使用者進行變更，不會立即套用變更。若要套用變更，您必須等待下一個維護時段或重新啟動代理程式。如需更多詳細資訊，請參閱 Amazon MQ 代理程式組態生命週期。

下圖說明組態生命週期。

> **Important**
>
> 下一個排定的維護時段會觸發重新啟動。如果代理程式在下一個排定的維護時段之前重新啟動，則會在重新啟動後套用變更。




針對 ActiveMQ，組態使用 XML 格式 (類似於 ActiveMQ 的 `activemq.xml` 檔案) 且包含代理程式的所有設定。如需建立、套用和編輯 ActiveMQ 代理程式組態的詳細資訊，請參閱 Creating and applying broker configurations。

針對 RabbitMQ，組態使用 Cuttlefish 格式且包含代理程式的所有設定。如需建立、套用和編輯 RabbitMQ 代理程式組態的詳細資訊，請參閱 Creating and applying broker configurations。

# 執行個體類型

代理程式執行個體類別 (m5、t3) 和大小 (large、micro) 的合併說明是代理程式執行個體類型 (例如，mq.m5.large)。下表列出每種支援的引擎類型可用的 Amazon MQ 代理程式執行個體類型。

主題



## Amazon MQ for ActiveMQ 執行個體類型

> **Important**
>
> 您只能將 Amazon EBS 搭配 mq.m5 代理程式執行個體類型系列使用。如需詳細資訊，請參閱 Storage。

| 執行個體類型 | vCPU | 記憶體 (GiB) | 網路效能 | 備註 |
| --- | --- | --- | --- | --- |
| mq.t2.micro | 1 | 1 | 低 | 使用 mq.t2.micro 執行個體類型進行 Amazon MQ 的基本評估。此執行個體類型 (僅限單一執行個體代理程式) 符合 AWS 免費方案的資格。<br><br>**Note**<br>mq.t2.micro 執行個體類型的使用受限於 |

| 執行個體類型 | vCPU | 記憶體 (GiB) | 網路效能 | 備註 |
| --- | --- | --- | --- | --- |
| | | | | CPU 額度和基準效能—並能夠大幅提升 效能以超越基準水準 (如需詳細資訊，請參閱 CpuCreditBalance 指標)。如果您的應用程式需要固定效能，請考慮使用 mq.m5.large 執行個體類型。 |
| mq.t3.micro | 2 | 1 | 低 | 使用 mq.t3.micro 執行個體類型進行 Amazon MQ 的基本評估。此執行個體類型 (僅限單一執行個體代理程式) 符合 AWS 免費方案的資格。 |

| 執行個體類型 | vCPU | 記憶體 (GiB) | 網路效能 | 備註 |
| --- | --- | --- | --- | --- |
| mq.m4.large | 2 | 8 | 適中 | 將 mq.m4.large 執行個體類型用於現有代理程式部署的相容性。建議您將 mq.m5.* 執行個體用於新的代理程式。 |
| mq.m5.large | 2 | 8 | 高 | 將 mq.m5.large 執行個體用於一般開發、測試及生產工作負載。 |

| 執行個體類型 | vCPU | 記憶體 (GiB) | 網路效能 | 備註 |
| --- | --- | --- | --- | --- |
| mq.m5.xlarge | 4 | 16 | 高 | 將 mq.m5.xlarge 、mq.m5.2xlarge 和 mq.m5.4xlarge 執行個體類型用於一般開發、測試，以及需要高輸送量的生產工作負載。<br><br>**Note**<br>當您的系統使用持久性訊息時，其傳輸量取決於訊息的耗用速度。如果未立即耗用訊息，則使用大型執行個體類型搭配持久性訊息可能不會提高系統傳輸量。在此情況下，我們建議您將 concurrentStoreAnd |
| mq.m5.2xlarge | 8 | 32 | 高 | |
| mq.m5.4xlarge | 16 | 64 | 高 | |

| 執行個體類型 | vCPU | 記憶體 (GiB) | 網路效能 | 備註 |
| --- | --- | --- | --- | --- |
| | | | | DispatchQueues 屬性設定為 false。如需詳細資訊，請參閱 對於具有緩慢消費者的佇列，停用並行存放和分派。 |

如需傳輸量考量的詳細資訊，請參閱為最佳傳輸量選擇正確的代理程式執行個體類型。

## Amazon MQ for RabbitMQ 執行個體類型

**Important**

您不能將代理程式從 mq.m5. 執行個體類型降級至 mq.t3.micro 執行個體類型。

| 執行個體類型 | vCPU | 記憶體 (GiB) | 網路效能 | 備註 |
| --- | --- | --- | --- | --- |
| mq.t3.micro | 2 | 1 | 低 | 使用 mq.t3.micro 執行個體類型進行 Amazon MQ 的基本評估。此執行個體類型 (僅限單一執行個體代理程式) |

| 執行個體類型 | vCPU | 記憶體 (GiB) | 網路效能 | 備註 |
| --- | --- | --- | --- | --- |
| | | | | 符合 AWS 免費方案的資格。<br><br>**Important**<br>mq.t3.micro 執行個體類型不支援叢集部署。 |
| mq.m5.large | 2 | 8 | 高 | 將 mq.m5.large 執行個體用於一般開發、測試及生產工作負載。 |
| mq.m5.xlarge | 4 | 16 | 高 | 將 mq.m5.xlarge 、mq.m5.2xlarge 和 mq.m5.4xlarge 執行個體類型用於一般開發、測試，以及需要高輸送量的生產工作負載。 |
| mq.m5.2xlarge | 8 | 32 | 高 | |
| mq.m5.4xlarge | 16 | 64 | 高 | |

# 標記 資源

Amazon MQ 支援資源標記，以協助您追蹤成本分配。您可以透過建立資源或檢視資源的詳細資訊，來為其加上標籤。

主題

- 進行標記以分配成本

- 在 Amazon MQ 主控台中管理標籤
- 使用 Amazon MQ API 動作來進行管理

## 進行標記以分配成本

若要整理和辨識 Amazon MQ 資源，以分配成本，您可新增中繼資料標籤來識別代理程式或組態的用途。當您擁有許多代理程式時，這項功能尤其實用。您可以使用成本分配標籤來整理您的 AWS 帳單，以反映您自己的成本結構。若要這麼做，請註冊取得 AWS 帳戶帳單，以納入標籤索引鍵和鍵值。如需詳細資訊，請參閱 AWS Billing 使用者指南中的設定每月成本分配報告。

例如，您可以新增標籤，來代表成本中心和 Amazon MQ 資源的用途：

| 資源 | 金鑰 | 數值 |
| --- | --- | --- |
| Broker1 | Cost Center | 34567 |
| | Stack | Production |
| Broker2 | Cost Center | 34567 |
| | Stack | Production |
| Broker3 | Cost Center | 12345 |
| | Stack | Development |

這種標記機制，可讓您將同一個成本中心內執行相關任務的兩個代理程式歸為同一組，並使用不同的成本分配標籤來標記不相關的代理程式。

## 在 Amazon MQ 主控台中管理標籤

### 為新資源加上標籤

Amazon MQ 可讓您在建立資源時為其加上標籤。您可以針對自己正在 Amazon MQ 主控台中所建立的資源，為其加上標籤。

在建立新的代理程式時新增標籤：

1. 從 Create a broker (建立代理程式) 頁面，選擇 Additional settings (其他設定)。
2. 在 Tags (標籤) 中，選擇 Add tag (新增標籤)。
3. 輸入 Key (索引鍵) 和 Value (值) 的對組。


4. (選用) 選擇 Add tag (新增標籤)，來為您的代理程式加上多個標籤。
5. 選擇 Create broker (建立代理程式)。

在建立組態時加上標籤：

1. 從 Create configuration (建立組態) 頁面，選擇 Advanced (進階)。
2. 在 Create configuration (建立組態) 頁面的 (Tags) 標籤中，選擇 Add tag (新增標籤)。
3. 輸入 Key (索引鍵) 和 Value (值) 的對組。
4. (選用) 選擇 Add tag (新增標籤)，來為您的組態加上多個標籤。
5. 選擇 Create configuration (建立組態)。

## 檢視與管理現有資源的標籤

Amazon MQ 可讓您在 Amazon MQ 主控台中檢視和管理資源的標籤。藉由在某個資源的詳細資訊頁面上編輯標籤，您可以管理個別資源的標籤。若要編輯 Amazon MQ 資源的標籤：

1. 在 Amazon MQ 主控台中選取代理程式或組態。

   在 Tags (標籤) 區塊中，檢閱該資源的現有標籤。
2. 若要新增或管理現有的標籤，請選擇 Edit (編輯) (或 Create tag (建立標籤) - 如果沒有現有的標籤)。
3. 更新資源的標籤：

   - 若要修改現存標籤，請編輯Key (索引鍵) 和 Value (值)。

- 若要移除現有的標籤，請選擇 Remove (移除)。
- 若要新增標籤，請選擇 Add tag (新增標籤)，然後輸入 Key (索引鍵) 和 Value (值)。

4. 選取 Save (儲存)。

## 使用 Amazon MQ API 動作來進行管理

Amazon MQ 可讓您使用 REST API 來檢視和管理資源的標籤。

如需詳細資訊，請參閱 Amazon MQ REST API 參考。

# 使用 Amazon MQ to ActiveMQ

Amazon MQ 可讓您利用符合您需求的運算和儲存資源輕鬆地建立訊息代理程式。您可以使用 AWS Management Console、 Amazon MQ REST API 或 AWS Command Line Interface 來建立、管理和刪除代理程式。

本節描述 ActiveMQ 和 RabbitMQ 引擎類型之訊息代理程式的基本元素、列出可用的 Amazon MQ 代理程式執行個體類型及其狀態，並提供代理程式架構和組態選項的概觀。

若要了解 Amazon MQ REST API，請參閱 Amazon MQ REST API 參考。

主題



## ActiveMQ 引擎

本節描述 ActiveMQ 代理程式的基本元素、提供 ActiveMQ 代理程式架構選項的概觀、說明代理程式組態參數，以及提供使用 Java Message Service (JMS) 的運作範例。

主題



### 基本元素

本節簡介了解 Amazon MQ 上 ActiveMQ 所需的重要概念。

主題

- 中介裝置
- 中介裝置執行個體類型
- 組態
- 使用者
- Storage

## 中介裝置

代理程式是在 Amazon MQ 上執行的訊息代理程式環境。這是 Amazon MQ 的基本建置區塊。代理程式執行個體類別 (`m5`、`t3`) 和大小 (`large`、`micro`) 的合併說明是代理程式執行個體類型 (例如，`mq.m5.large`)。如需更多詳細資訊，請參閱 Broker instance types。

- 單一執行個體代理程式是由一個可用區域中的一個代理程式組成。代理程式會與您的應用程式以及 Amazon EBS 或 Amazon EFS 儲存磁碟區進行通訊。
- 作用中/待命代理程式是由兩個不同可用區域中的兩個代理程式所組成，並設定於備援組合中。這些代理程式會與您的應用程式及 Amazon EFS 同步通訊。

如需詳細資訊，請參閱 Broker Architecture。

當 Apache 發佈新版本時，您可以啟用自動次要版本升級，以升級到代理程式引擎的新次要版本。自動升級會發生於由星期幾、一天中的時間 (24 小時制) 和時區 (預設為 UTC) 所定義的維護時段期間。

如需建立和管理代理程式的詳細資訊，請參閱以下各節：

- Creating and configuring a broker
- 中介裝置
- Broker statuses

### 支援的線路通訊協定

您可以存取代理程式，方法為使用 ActiveMQ 支援的任何程式設計語言，並明確地為下列通訊協定啟用 TLS：

- AMQP
- MQTT
- MQTT over WebSocket

- OpenWire
- STOMP
- STOMP over WebSocket

### 屬性

ActiveMQ 代理程式具有多個屬性，例如：

- 名稱 (`MyBroker`)
- ID (`b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9`)
- Amazon 資源名稱 (ARN) (`arn:aws:mq:us-east-2:123456789012:broker:MyBroker:b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9`)
- ActiveMQ 網頁主控台 URL (`https://b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-east-2.amazonaws.com:8162`)

  如需詳細資訊，請參閱 Apache ActiveMQ 文件中的 Web 主控台。

  > **Important**
  >
  > 如果您指定的授權映射不包含 `activemq-webconsole` 群組，您便無法使用 ActiveMQ Web 主控台，因為該群組未獲授權傳送或接收來自 Amazon MQ 代理程式的訊息。

- 線路通訊協定端點：
  - `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**
    >
    > 這是 OpenWire 端點。

  - `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`

如需詳細資訊，請參閱 Apache ActiveMQ 文件中的設定傳輸。

> **Note**
>
> 對於作用中/待命代理程式，Amazon MQ 會提供兩個 ActiveMQ Web 主控台 URL，但一次只有一個作用中的 URL。同樣地，Amazon MQ 為每個線路通訊協定提供兩個端點，但每個配對中一次只有一個作用中的端點。`-1` 和 `-2` 尾碼表示備援組合。

如需代理程式屬性的完整清單，請參閱 Amazon MQ REST API 參考中的以下各節：

- REST 操作 ID：中介裝置
- REST 操作 ID：中介裝置
- REST 操作 ID：中介裝置重新開機

## 中介裝置執行個體類型

> **Important**
>
> 您只能將 Amazon EBS 搭配 `mq.m5` 代理程式執行個體類型系列使用。如需詳細資訊，請參閱 Storage。

| 執行個體類型 | vCPU | 記憶體 (GiB) | 網路效能 | 備註 |
| --- | --- | --- | --- | --- |
| `mq.t2.micro` | 1 | 1 | 低 | 使用 `mq.t2.micro` 執行個體類型進行 Amazon MQ 的基本評估。此執行個體類型 (僅限單一執行個體代理程式) |

| 執行個體類型 | vCPU | 記憶體 (GiB) | 網路效能 | 備註 |
| --- | --- | --- | --- | --- |
| | | | | 符合 AWS 免費方案的資格。<br><br>**Note**<br>`mq.t2.micro` 執行個體類型的使用受限於 CPU 額度和基準效能—並能夠大幅提升 效能以超越基準水準 (如需詳細資訊，請參閱 `CpuCreditBalance` 指標)。如果您的應用程式需要固定效能，請考慮使用 `mq.m5.large` 執行個體類型。 |

| 執行個體類型 | vCPU | 記憶體 (GiB) | 網路效能 | 備註 |
| --- | --- | --- | --- | --- |
| mq.t3.micro | 2 | 1 | 低 | 使用 mq.t3.micro 執行個體類型進行 Amazon MQ 的基本評估。此執行個體類型 (僅限單一執行個體代理程式) 符合 AWS 免費方案的資格。 |
| mq.m4.large | 2 | 8 | 適中 | 將 mq.m4.large 執行個體類型用於現有代理程式部署的相容性。建議您將 mq.m5.* 執行個體用於新的代理程式。 |
| mq.m5.large | 2 | 8 | 高 | 將 mq.m5.large 執行個體用於一般開發、測試及生產工作負載。 |

| 執行個體類型 | vCPU | 記憶體 (GiB) | 網路效能 | 備註 |
| --- | --- | --- | --- | --- |
| mq.m5.xlarge | 4 | 16 | 高 | 將 mq.m5.xlarge 、mq.m5.2xlarge 和 mq.m5.4xlarge 執行個體類型用於一般開發、測試，以及需要高輸送量的生產工作負載。<br><br>**Note**<br>當您的系統使用持久性訊息時，其傳輸量取決於訊息的耗用速度。如果未立即耗用訊息，則使用大型執行個體類型搭配持久性訊息可能不會提高系統傳輸量。在此情況下，我們建議您將 concurrentStoreAnd |
| mq.m5.2xlarge | 8 | 32 | 高 | |
| mq.m5.4xlarge | 16 | 64 | 高 | |

| 執行個體類型 | vCPU | 記憶體 (GiB) | 網路效能 | 備註 |
| --- | --- | --- | --- | --- |
| | | | | `DispatchQueues` 屬性設定為 `false`。如需更多詳細資訊，請參閱對於具有緩慢消費者的佇列，停用並行存放和分派。 |

如需傳輸量考量的詳細資訊，請參閱為最佳傳輸量選擇正確的代理程式執行個體類型。

## 組態

組態以 XML 格式 (類似於 ActiveMQ 的 `activemq.xml` 檔案) 包含所有的 ActiveMQ 代理程式設定。您可以在建立任何代理程式之前建立組態。然後，您可以將組態套用至一或多個代理程式。

> **Important**
>
> 對組態進行變更，不會立即將變更套用至代理程式。若要套用變更，您必須等待下一個維護時段或重新啟動代理程式。如需更多詳細資訊，請參閱 Amazon MQ 代理程式組態生命週期。您目前無法刪除組態。

如需建立、編輯和管理組態的詳細資訊，請參閱以下各節：

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

若要追蹤您對組態做出的變更，您可以建立組態修訂。如需更多詳細資訊，請參閱 Creating and applying broker configurations。

屬性

代理程式組態具有多個屬性，例如：

- 名稱 (MyConfiguration)
- ID (c-1234a5b6-78cd-901e-2fgh-3i45j6k178l9)
- Amazon 資源名稱 (ARN) (arn:aws:mq:us-east-2:123456789012:configuration:c-1234a5b6-78cd-901e-2fgh-3i45j6k178l9)

如需組態屬性的完整清單，請參閱 Amazon MQ REST API 參考中的以下各節：

- REST 操作 ID：組態
- REST 操作 ID：組態

如需組態修訂屬性的完整清單，請參閱以下各節：

- REST 操作 ID：組態修訂
- REST 操作 ID：組態修訂

## 使用者

ActiveMQ 使用者是可以存取 ActiveMQ 代理程式的佇列和主題的人員或應用程式。您可以將使用者設定為具有特定許可。例如，您可以允許某些使用者存取 ActiveMQ Web 主控台。

群組是一個語義標籤。您可以將群組指派給使用者，並設定可供群組傳送、接收和管理特定佇列和主題的許可。

> **Important**
>
> 對使用者進行變更，不會立即將變更套用至使用者。若要套用變更，您必須等待下一個維護時段或重新啟動代理程式。如需更多詳細資訊，請參閱 Amazon MQ 代理程式組態生命週期。

如需使用者和群組的詳細資訊，請參閱 Apache ActiveMQ 文件中的以下章節：

- 授權
- 授權範例

如需建立、編輯和刪除 ActiveMQ 使用者的詳細資訊，請參閱以下各節：

- 建立和管理 ActiveMQ 代理程式使用者
- 使用者

#### 屬性

如需使用者屬性的完整清單，請參閱 Amazon MQ REST API 參考中的以下各節：

- REST 操作 ID：使用者
- REST 操作 ID：使用者

### Storage

Amazon MQ for ActiveMQ 支援 Amazon Elastic File System (EFS) 和 Amazon Elastic Block Store (EBS)。根據預設，ActiveMQ 代理程式使用 Amazon EFS 進行代理程式儲存。若要利用跨多個可用區域的高耐久性和複寫功能，請使用 Amazon EFS。若要利用低延遲和高輸送量，請使用 Amazon EBS。

> **Important**
>
> - 您只能將 Amazon EBS 搭配 `mq.m5` 代理程式執行個體類型系列使用。
> - 雖然您可以變更代理程式執行個體類型，但無法在建立代理程式後變更代理程式儲存類型。
> - Amazon EBS 會複寫單一可用區域內的資料，且不支援 ActiveMQ 作用中/待命部署模式。

#### 儲存類型之間的差異

下表提供 ActiveMQ 代理程式的記憶體內、Amazon EFS 和 Amazon EBS 儲存類型之間差異的簡要概觀。

| 儲存類型 | Persistence | 範例使用案例 | 每個生產者每秒排入佇列的訊息數目大約上限 (1KB 訊息) | 複寫 |
|---|---|---|---|---|
| 記憶體內 | 非持續性 | • 股票報價<br>• 位置資料更新<br>• 經常變更的資料 | 5,000 | 無 |
| Amazon EBS | 持續 | • 大量文字<br>• 訂單處理 | 500 | 單一可用區域 (AZ) 內的多個複本 |
| Amazon EFS | 持續 | 金融交易 | 80 | 跨多個 AZ 的多個複本 |

記憶體內訊息儲存提供最低延遲和最高輸送量。不過，訊息在執行個體取代或代理程式重新啟動期間會遺失。

Amazon EFS 設計成高耐用性，可跨多個 AZ 複寫，以避免因任何單一元件故障或影響 AZ 可用性的問題而造成資料遺失。Amazon EBS 已針對輸送量進行最佳化處理，並且在單一 AZ 內的多部伺服器之間進行複寫。

## 代理程式架構

您可以建立 Amazon MQ for ActiveMQ 代理程式做為單一執行個體代理程式或作用中/待命代理程式。對於這兩種部署模式，Amazon MQ 會經由備援方式儲存資料，以提供高耐久性。

> **Note**
>
> Amazon MQ 使用 Apache KahaDB 做為其資料存放區。不支援其他資料存放區，例如 JDBC 和 LevelDB。

您可以存取代理程式，方法為使用 ActiveMQ 支援的任何程式設計語言，並明確地為下列通訊協定啟用 TLS：

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

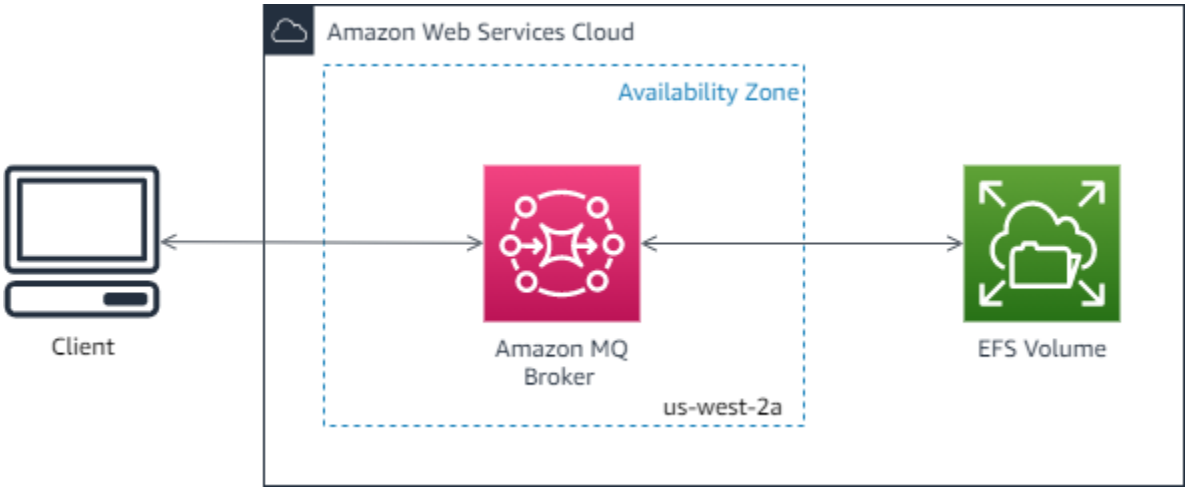
主題

- Amazon MQ 單一執行個體代理程式
- 高可用性的 Amazon MQ 作用中/待命代理程式
- Amazon MQ 代理程式網路

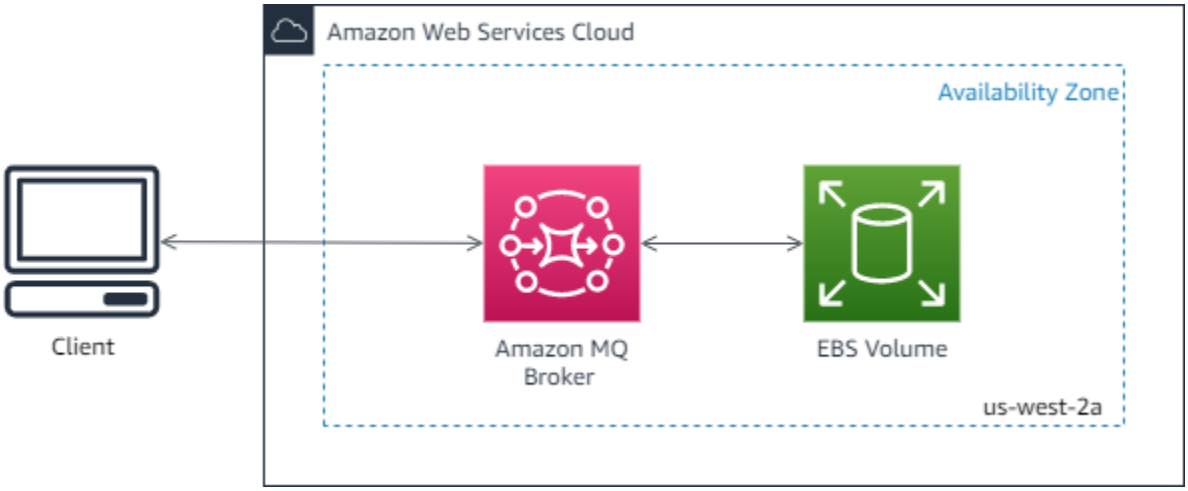
## Amazon MQ 單一執行個體代理程式

單一執行個體代理程式是由一個可用區域中的一個代理程式組成。代理程式會與您的應用程式以及 Amazon EBS 或 Amazon EFS 儲存磁碟區進行通訊。Amazon EFS 儲存磁碟區的設計訴求是要跨多個可用區域 (AZ) 存放資料，以提供最高層級的耐久性和可用性。Amazon EBS 提供針對低延遲和高輸送量最佳化的區塊層級儲存。如需儲存選項的詳細資訊，請參閱 Storage。

下圖說明單一執行個體代理程式，具有跨多個 AZ 複寫的 Amazon EFS 儲存。




下圖說明單一執行個體代理程式，具有在單一 AZ 內多部伺服器之間複寫的 Amazon EBS 儲存。



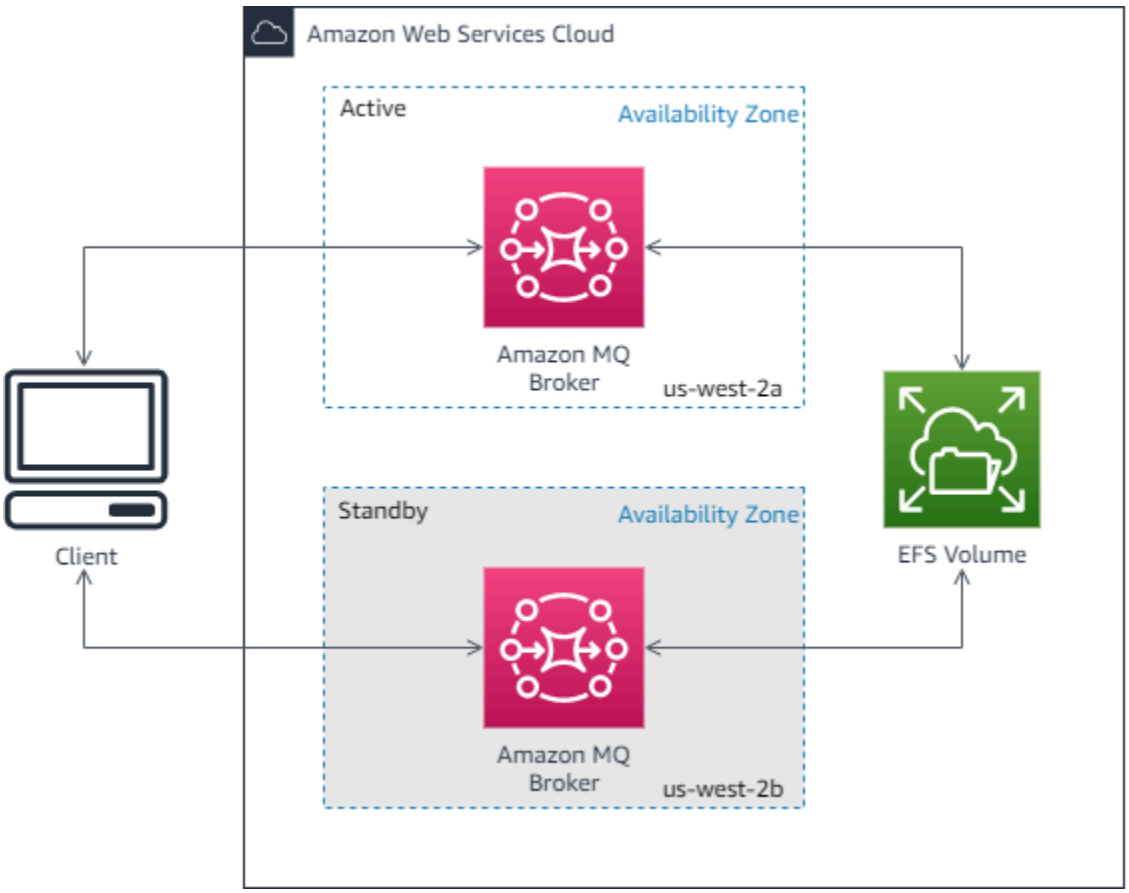

## 高可用性的 Amazon MQ 作用中/待命代理程式

作用中/待命代理程式是由兩個不同可用區域中的兩個代理程式所組成，並設定於備援組合中。這些代理程式會與您的應用程式及 Amazon EFS 同步通訊。Amazon EFS 儲存磁碟區的設計訴求是要跨多個可用區域 (AZ) 存放資料，以提供最高層級的耐久性和可用性。如需更多詳細資訊，請參閱 Storage。

通常，代理程式執行個體中，只有一個是隨時作用中，而另外一個則處於待命中。如果其中一個代理程式執行個體發生故障或進行維護，Amazon MQ 需要一段時間才能將非作用中執行個體停止服務。這可讓狀況良好的待命執行個體變成作用中，並開始接受傳入的通訊。當您重新啟動代理程式時，容錯移轉只需要幾秒鐘的時間。

對於作用中/待命代理程式，Amazon MQ 會提供兩個 ActiveMQ Web 主控台 URL，但一次只有一個作用中的 URL。同樣地，Amazon MQ 為每個線路通訊協定提供兩個端點，但每個配對中一次只有一個作用中的端點。`-1` 和 `-2` 尾碼表示備援組合。對於線路通訊協定端點，您可以允許應用程式使用容錯移轉傳輸連線到任一端點。

下圖說明作用中/待命代理程式，具有跨多個 AZ 複寫的 Amazon EFS 儲存。




## Amazon MQ 代理程式網路

Amazon MQ 支援 ActiveMQ 的代理程式網路功能。

代理程式網路由多個同時作用中單一執行個體代理程式或作用中/待命代理程式組成。您可以在各種拓撲 (例如，集中器、中樞和支點、樹狀結構或網格) 中設定代理程式網路，這取決於您的應用程式的需求，例如高可用性和可擴展性。例如，中樞和支點代理程式網路網路可以提高恢復能力，如果一個代理程式無法連線，則會保留訊息。具有集中器拓撲的代理程式網路可以從接受傳入訊息的大量代理程式收集訊息，並將它們集中到更多中央代理程式，以便更妥善處理許多傳入訊息的載入。

如需教學課程和詳細的組態資訊，請參閱下列內容：

- Creating and Configuring a Network of Brokers
- 正確地設定您的代理程式網路
- `networkConnector`
- `networkConnectionStartAsync`
- ActiveMQ 文件中的 Networks of Brokers (代理程式網路)

下列是使用代理程式網路的優點：

- 建立代理程式網路可讓您透過新增代理程式執行個體，來提高彙總的輸送量，並達到最大的生產者與使用者連線數。
- 您可以讓生產者和使用者注意到多個運作中的代理程式執行個體，來確保更高的可用性。如果生產者和使用者目前所連線的執行個體變得無法使用，這項機制就能讓這些人重新連線到新的執行個體。
- 由於生產者和使用者可以立即重新連線到代理程式網路中的另一個節點，而且不需等待備用的代理程式執行個體提升，因此代理程式網路內的用戶端重新連線，速度會比運作中/待命的代理程式快，提供高可用性。

主題

- 代理程式網路的運作方式？
- 代理程式網路如何處理登入資料？
- 藍圖範例
- 代理程式網路拓撲
- 跨區域
- 搭配傳輸連接器的動態容錯移轉

## 代理程式網路的運作方式？

Amazon MQ 透過多種方式來支援 ActiveMQ 代理程式網路功能。首先，您可以編輯每個代理程式組態中的參數，來建立代理程式網路，就如同使用原生 ActiveMQ 時所能進行的動作。其次，Amazon MQ 有範例藍圖，可使用 AWS CloudFormation 來自動建立代理程式網路。您可以直接從 Amazon MQ 主控台來部署這些範例藍圖，或是您可編輯相關的 AWS CloudFormation 範本以建立自己的拓撲和組態。

若要建立代理程式網路，可使用網路連接器，將一個代理程式連接到另一個。連線後，這些代理程式會提供訊息轉傳功能。例如，如果 Broker1 建立連接到 Broker2 的網路連接器，則當該代理程式上存在佇列或主題的取用者時，Broker1 上的訊息會轉傳到 Broker2。如果網路連接器是設定為 `duplex`，則訊息也會從 Broker2 轉傳到 Broker1。網路連接器是在代理程式的組態中設定。請參閱 組態。例如，下面是代理程式組態中的範例 `networkConnector` 項目：

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

```
</networkConnectors>
```

代理程式網路可確保訊息從一個代理程式執行個體傳送到另一個，只將訊息轉傳給具有對應使用者的代理程式執行個體。至於網路內相鄰代理程式執行個體的優點，在於 ActiveMQ 會傳送訊息到諮詢主題，此主題是關於連線到網路和從網路中斷的生產者與使用者。當代理程式執行個體接收到關於取用者的資訊，而此取用者是從特定的目的地使用時，則代理程式執行個體會開始轉傳訊息。如需詳細資訊，請參閱 ActiveMQ 文件中的 Advisory Topics (諮詢主題)。

代理程式網路如何處理登入資料？

網路中的代理程式 A 若要連線到代理程式 B，代理程式 A 必須使用有效的登入資料，就如同其他任何生產者或使用者。您必須先在代理程式 A 上建立使用者，其值與代理程式 B 上的另一個使用者相同 (這些是共用相同使用者名稱和密碼值的獨立、唯一使用者)，而不是在代理程式 A 的 `<networkConnector>` 組態中提供密碼。當您在 `<networkConnector>` 組態中指定 `userName` 屬性時，Amazon MQ 將會在執行時間自動新增密碼。

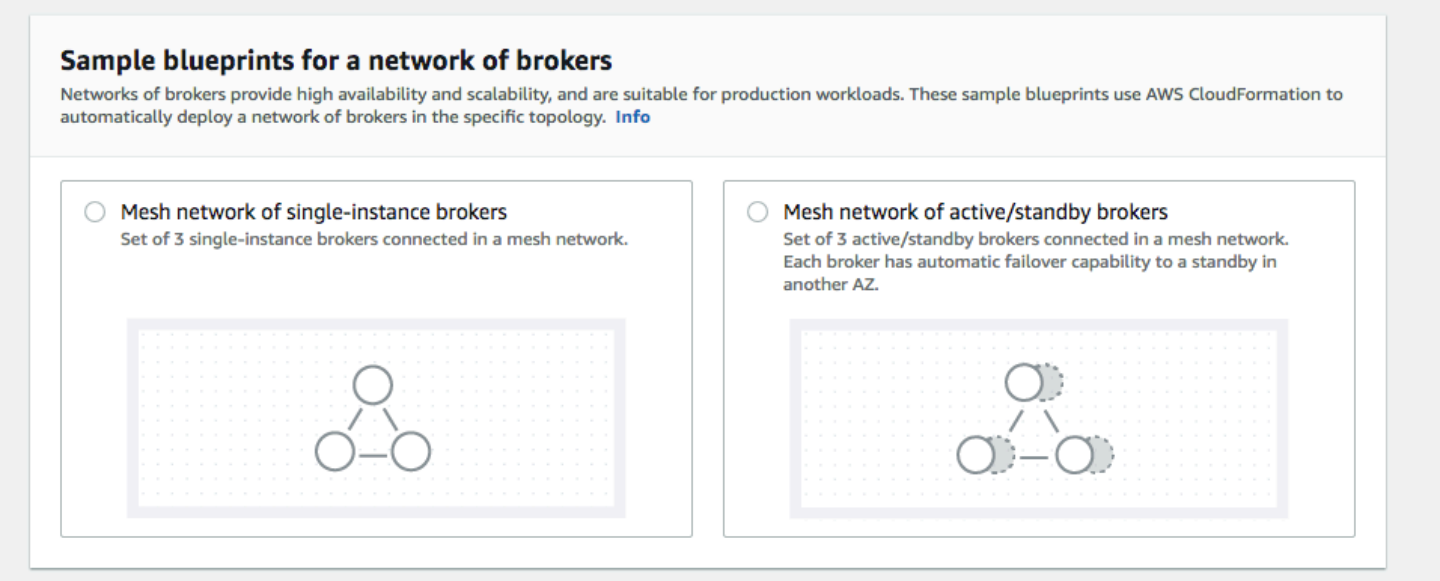
> **Important**
>
> 不要指定 `<networkConnector>` 的 `password` 屬性。我們不建議在代理程式的組態檔案中儲存純文字密碼，因為這樣在 Amazon MQ 主控台中就看得到密碼。如需更多詳細資訊，請參閱 Configure Network Connectors for Your Broker。

代理程式必須位於同一個 VPC 或對等 VPC 中。如需詳細資訊，請參閱 Creating and Configuring a Network of Brokers 教學課程中的 先決條件。

藍圖範例

若要開始使用代理程式網路，Amazon MQ 提供了藍圖範例。這些藍圖範例使用 AWS CloudFormation 建立了代理程式網路部署和所有相關資源。您可使用下列兩種藍圖範例：

1. 單一執行個體代理程式的網狀網路
2. 運作中/待命代理程式的網狀網路




從 Create brokers (建立代理程式) 頁面，選取其中一個藍圖範例，然後選擇 Next (下一步)。資源建立之後，請在 Amazon MQ 主控台中，檢閱所產生的代理程式及其組態。

藉由在代理程式組態中，建立代理程式和設定不同的 `networkConnector` 元素，您可以使用多種不同的拓撲，來建立代理程式的網路。如需關於設定代理程式網路的詳細資訊，請參閱 ActiveMQ 文件中的 Networks of Brokers (代理程式網路)。

代理程式網路拓撲

透過部署代理程式，然後在其組態中設定 `networkConnector` 項目，您可以使用不同的網路拓撲來建置代理程式網路。網路連接器提供在互連的代理程式之間，轉傳隨需訊息的功能。連線可設定為雙工或非雙工，前者會在代理程式之間雙向轉傳訊息，後者指會將訊息從一個代理程式傳佈到另一個。例如，如果在 Broker1 和 Broker2 之間具有雙工連線，則出現使用者時，訊息會在代理程式之間彼此轉傳。


如果使用雙工網路連接器，訊息會從每個代理程式轉傳到其他代理程式。這些是隨需轉傳：如果 Broker2 上存在 Broker1 上訊息的使用者，則會轉傳訊息。同樣地，如果 Broker1 上存在 Broker2 上訊息的使用者，也會轉傳訊息。

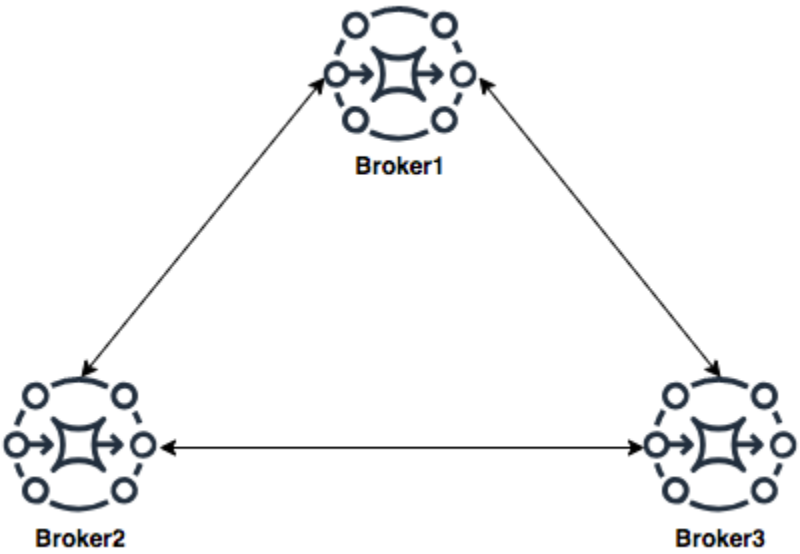
如果是非雙工連線，則訊息只會從一個代理程式轉傳到其他代理程式。在此範例中，如果 Broker2 上存在 Broker1 上訊息的使用者，則會轉傳訊息。但訊息將不會從 Broker2 轉傳到 Broker1。


使用這兩種雙工和非雙工網路連接器，就有可能以任何數量規模的網路拓撲，來建置代理程式的網路。

> **Note**
>
> 在每個網路拓撲範例中，`networkConnector` 元素會參考自己所連接代理程式的端點。將 `uri` 屬性中的代理程式端點項目，換成您代理程式的端點。請參閱 Listing brokers and viewing broker details。

網狀拓撲

網狀拓撲提供彼此連線的多個代理程式。這個簡單的範例連接了三個單一執行個體代理程式，但您可以設定更多代理程式成為網狀網路。



這個拓撲和內含多對作用中/待命代理程式所構成的網狀拓撲，都可以在 Amazon MQ 主控台中建立使用藍圖範例來建立。您可以建立這些藍圖範例，來查看正常運作的代理程式網路，並檢閱這些代理程式的設定方式。

您可以透過將網路連接器加到 Broker1 (該網路連接器會在 Broker2 和 Broker3 之間進行雙工連線，以及在 Broker2 和 Broker3 之間進行單一雙工連線)，來設定三個代理程式網狀網路。

Broker1 的網路連接器：

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

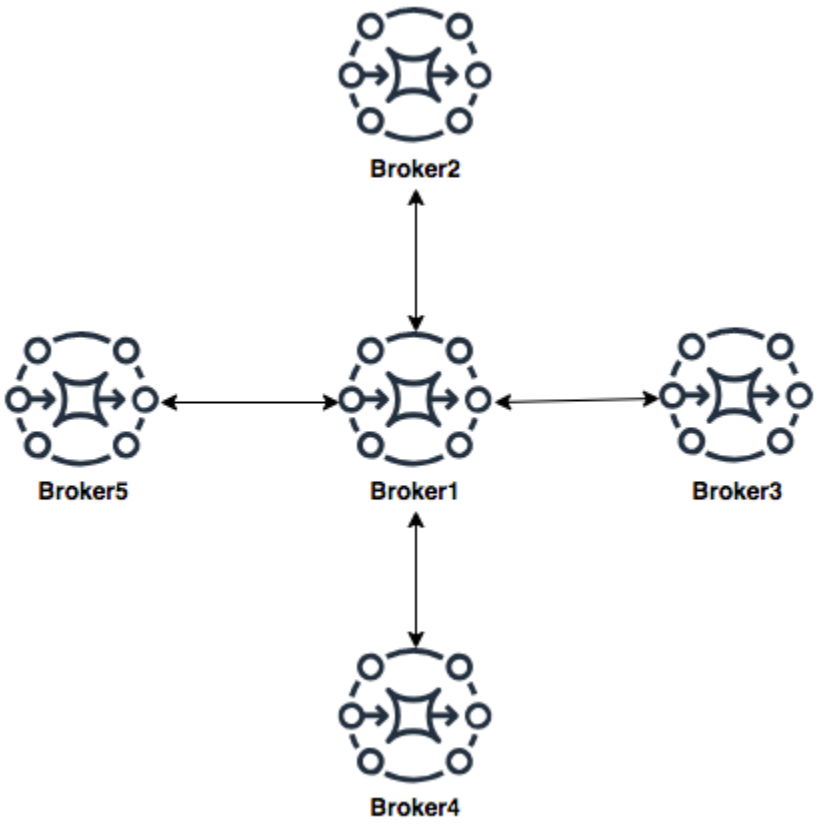
Broker2 的網路連接器：

```
<networkConnectors>
    <networkConnector name="connector_2_to_3" userName="myCommonUser" duplex="true"
        uri="static:(ssl://b-743c885d-2244-4c95-af67-a85017ff234e-3.mq.us-
east-2.amazonaws.com:61617)"/>
</networkConnectors>
```

透過將上述的連接器加入 Broker1 and Broker2 的組態，您可以在這三個代理程式之間，建立網狀網路，來隨需在所有的代理程式之間轉傳訊息。如需更多詳細資訊，請參閱 Amazon MQ Broker Configuration Parameters。

中樞和支點拓撲

在中樞和支點拓撲中，如果支點上的任何代理程式有所中斷，將會保留訊息。訊息會轉傳到各處，而且只有中央的 Broker1 對網路的運作至關重要。



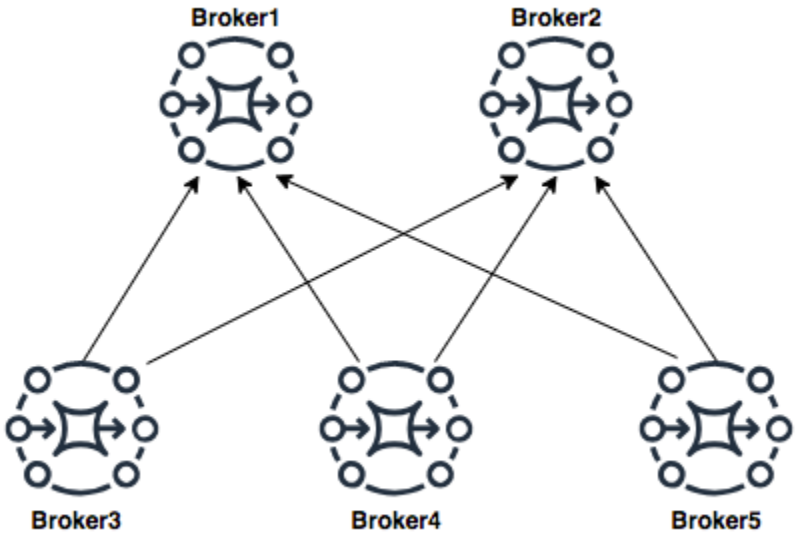

若要在此範例中設定代理程式的中樞和支點網路，您可以在 Broker1 的組態中將 networkConnector 新增至支點上的每個代理程式。

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

```
        uri="static:(ssl://b-62a7fb31-d51c-466a-a873-905cd660b553-4.mq.us-
east-2.amazonaws.com:61617)"/>
</networkConnectors>
```

集中器拓撲

在此拓撲範例中，位於底部的三個代理程式可以處理大量的連線，而這些訊息會集中到 Broker1 和 Broker2。其他每個代理程式，都具有非雙工連線，來連線到更多中心代理程式。若要擴展此拓撲的容量，您可以新增額外的代理程式，來接收訊息，並且將這些訊息集中到 Broker1 和 Broker2。




若要設定此種拓撲，位於底部的每個代理程式，都會包含網路連接器，連接到做為訊息集中目的地的每個代理程式。

Broker3 的網路連接器：

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

Broker4 的網路連接器：

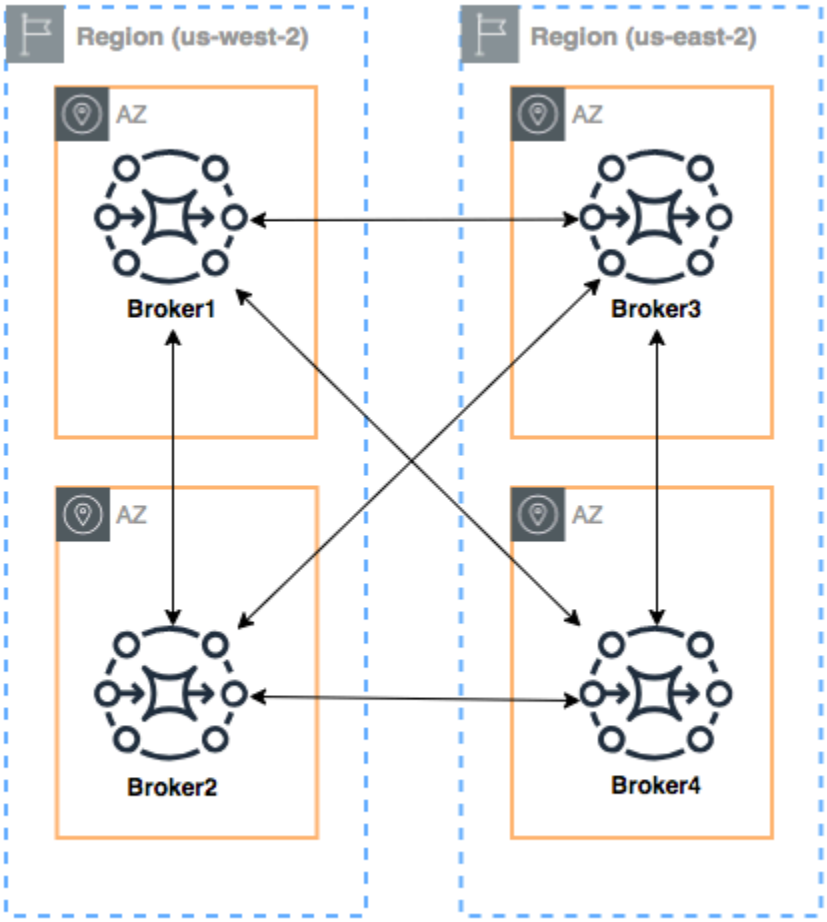
```
<networkConnectors>
    <networkConnector name="4_to_1" userName="myCommonUser" duplex="false"
        uri="static:(ssl://b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-
east-2.amazonaws.com:61617)"/>
    <networkConnector name="4_to_2" userName="myCommonUser" duplex="false"
        uri="static:(ssl://b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-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

Broker5 的網路連接器：

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

**跨區域**

若要設定跨 AWS 區域的代理程式網路，請在這些區域中部署代理程式，然後設定網路連接器連接至這些代理程式的端點。




若要像這個範例一樣設定代理程式網路，您可以將 networkConnectors 項目新增到 Broker1 和 Broker4 組態，以參考這些代理程式的線路層級端點 。

Broker1 的網路連接器：

```
<networkConnectors>
    <networkConnector name="1_to_2" userName="myCommonUser" duplex="true"
        uri="static:(ssl://b-9876l5k4-32ji-109h-8gfe-7d65c4b132a1-2.mq.us-
east-2.amazonaws.com:61617)"/>
    <networkConnector name="1_to_3" userName="myCommonUser" duplex="true"
        uri="static:(ssl://b-743c885d-2244-4c95-af67-a85017ff234e-3.mq.us-
east-2.amazonaws.com:61617)"/>
    <networkConnector name="1_to_4" userName="myCommonUser" duplex="true"
        uri="static:(ssl://b-62a7fb31-d51c-466a-a873-905cd660b553-4.mq.us-
east-2.amazonaws.com:61617)"/>
```

```
</networkConnectors>
```

Broker2 的網路連接器：

```
<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 的網路連接器：

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

搭配傳輸連接器的動態容錯移轉

除了設定 networkConnector 元素，您還可以將代理程式 transportConnector 選項設定成啟用動態容錯移轉，並在網路中新增或移除代理程式時重新平衡連線。

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

在此範例中，updateClusterClients 和 rebalanceClusterClients 都會設定為 true。在這種情況下，用戶端將會獲得網路內代理程式清單，並會在新的代理程式加入時要求重新平衡。

可用選項：

- updateClusterClients：將有關代理程式網路拓撲變更的資訊傳遞給用戶端。
- rebalanceClusterClients：將在代理程式網路中新增代理程式時，造成用戶端重新平衡所有的代理程式。

- `updateClusterClientsOnRemove`：在代理程式離開代理程式網路時，搭配拓撲資訊更新用戶端。

當 `updateClusterClients` 設為 True 時，用戶端即可設定為連接至代理程式網路中的單一代理程式。

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

當新的代理程式連線時，它將會收到網路中全部代理程式的 URI 清單。如果與代理程式的連線失敗，則連線會動態切換到已連線的其中一個代理程式。

如需關於容錯移轉的詳細資訊，請參閱 Active MQ 文件中的容錯移轉適用的代理程式選項。

## Amazon MQ for ActiveMQ 代理程式組態

組態使用 XML 格式 (類似於 ActiveMQ 的 `activemq.xml` 檔案)，包含 ActiveMQ 代理程式的所有設定。您可以在建立任何代理程式之前建立組態。然後，您可以將組態套用至一或多個代理程式。

主題

- 使用 Spring XML 組態檔案
- 建立、編輯和套用 ActiveMQ 代理程式組態
- Amazon MQ 組態中允許的元素
- Amazon MQ 組態中允許的元素及其屬性
- Amazon MQ 組態中允許的元素、子集合元素及其子元素

### 使用 Spring XML 組態檔案

系統會使用 Spring XML 檔案來設定 ActiveMQ 代理程式。您可以設定 ActiveMQ 代理程式的多個層面，例如預先定義的目的地、目的地政策、授權政策，以及外掛程式。Amazon MQ 會控制其中一些組態元素，例如網路傳輸和儲存。目前不支援其他組態選項，例如代理程式網路的建立。

在 Amazon MQ XML 結構描述中，指定了完整的一組支援組態選項。請使用下列連結下載支援結構描述的 zip 檔案。

- amazon-mq-active-mq-5.17.3.xsd.zip
- amazon-mq-active-mq-5.17.2.xsd.zip

- amazon-mq-active-mq-5.17.1.xsd.zip
- amazon-mq-active-mq-5.16.5.xsd.zip
- amazon-mq-active-mq-5.16.4.xsd.zip
- amazon-mq-active-mq-5.16.3.xsd.zip
- amazon-mq-active-mq-5.16.2.xsd.zip
- amazon-mq-active-mq-5.15.15.xsd.zip
- amazon-mq-active-mq-5.15.14.xsd.zip
- amazon-mq-active-mq-5.15.13.xsd.zip
- amazon-mq-active-mq-5.15.12.xsd.zip
- amazon-mq-active-mq-5.15.10.xsd.zip
- amazon-mq-active-mq-5.15.9.xsd.zip
- amazon-mq-active-mq-5.15.8.xsd.zip
- amazon-mq-active-mq-5.15.6.xsd.zip
- amazon-mq-active-mq-5.15.0.xsd.zip

您可以使用這些結構描述來驗證和清理組態檔案。Amazon MQ 也可讓您藉由上傳 XML 檔案來提供組態。當您上傳 XML 檔案時，Amazon MQ 會根據結構描述自動清除並移除無效及禁止的組態參數。

> **Note**
>
> 您只能對屬性使用靜態值。Amazon MQ 會從您的組態中清除包含 Spring 運算式、變數和元素參考的元素和屬性。

## 建立、編輯和套用 ActiveMQ 代理程式組態

組態以 XML 格式 (類似於 ActiveMQ 的 `activemq.xml` 檔案) 包含所有的 ActiveMQ 代理程式設定。您可以在建立任何代理程式之前建立組態。然後，您可以將組態套用至一或多個代理程式。您可以立即或在維護時段套用組態。

如需詳細資訊，請參閱下列內容：

- 組態
- Amazon MQ 代理程式組態生命週期
- Amazon MQ Broker Configuration Parameters

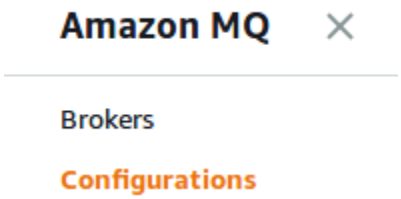
以下範例顯示如何使用 AWS Management Console 建立和套用 Amazon MQ 代理程式組態。

主題

- 建立新組態
- 建立新的組態修訂
- 將組態修訂套用至您的代理程式
- 編輯組態修訂

## 建立新組態

1. 登入 Amazon MQ 主控台。
2. 在左邊，展開導覽面板並選擇 Configurations (組態)。




3. 在 Configurations (組態) 頁面上，選擇 Create configuration (建立組態)。
4. 在 Create configuration (建立組態) 頁面的 Details (詳細資訊) 區段中，輸入 Configuration name (組態名稱) (例如 `MyConfiguration`)，然後選取 Broker engine (代理程式引擎) 版本。

> **Note**
>
> 若要進一步了解 Amazon MQ for ActiveMQ 支援的 ActiveMQ 引擎版本，請參閱 the section called “版本管理”。

5. 選擇 Create configuration (建立組態)。

## 建立新的組態修訂

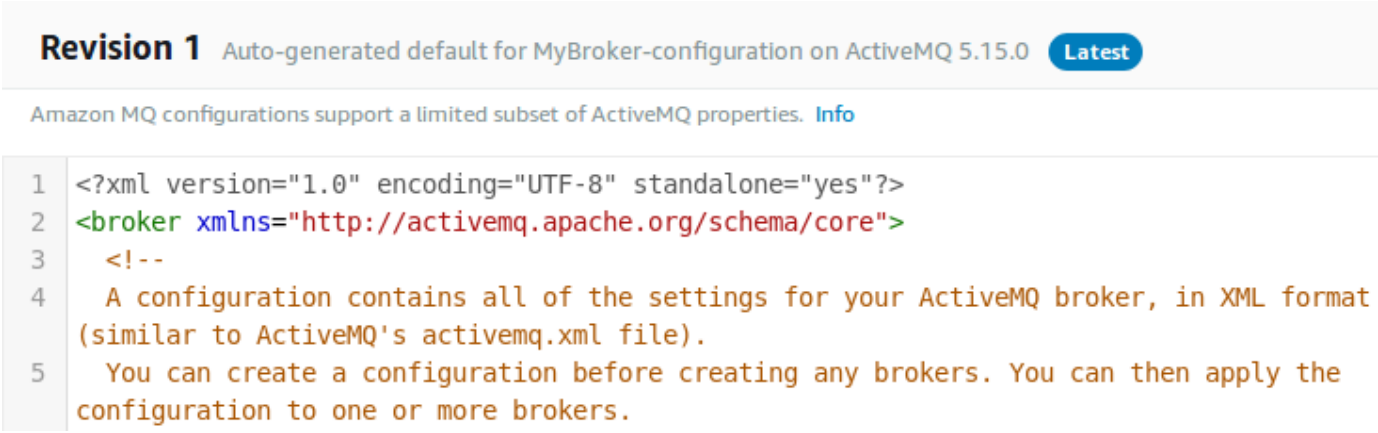
1. 從組態清單中，選擇 ***MyConfiguration***。

> **Note**
>
> 當 Amazon MQ 建立組態時，一律為您建立第一個組態修訂。

在 ***MyConfiguration (####)*** 頁面上，會顯示新組態版本所使用的代理程式引擎類型和版本 (例如 Apache ActiveMQ 5.15.8 (Apache ActiveMQ 5.15.8))。

2. 在 Configuration details (組態詳細資訊) 標籤上，會顯示組態修訂編號、描述及 XML 格式的代理程式組態。

> **Note**
>
> 編輯目前的組態會建立新的組態版本。




3. 選擇 Edit configuration (編輯組態)，然後變更 XML 組態。
4. 選擇 Save (儲存)。

Save revision (儲存修訂) 對話方塊隨即顯示。

5. (選用) 輸入 `A description of the changes in this revision`。
6. 選擇 Save (儲存)。

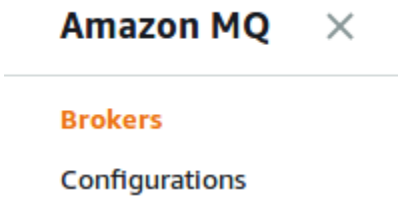
隨即儲存組態的新修訂版。

> **Important**
>
> Amazon MQ 主控台會根據結構描述，自動清理無效及禁止的組態參數。如需詳細資訊和允許的完整 XML 參數清單，請參閱 Amazon MQ Broker Configuration Parameters。對組態進行變更，不會立即將變更套用至代理程式。若要套用變更，您必須等待下一個維護時段或重新啟動代理程式。如需更多詳細資訊，請參閱 Amazon MQ 代理程式組態生命週期。

您目前無法刪除組態。

將組態修訂套用至您的代理程式

1. 在左邊，展開導覽面板並選擇 Brokers (代理程式)。




2. 從代理程式清單，選取您的代理程式 (例如 MyBroker)，然後選擇 Edit (編輯)。
3. 在 Edit ***MyBroker*** (編輯 MyBroker) 頁面的 Configuration (組態) 區段中，選取 Configuration (組態) 和 Revision (修訂)，然後選擇 Schedule Modifications (排程修改)。
4. 在 Schedule broker modifications (排定代理程式修改) 部分，選擇套用修改的時機是 During the next scheduled maintenance window (下一個排程的維護時段) 或 Immediately (立即)。

> ⚠ Important
>
> 您的代理程式會在重新啟動時離線。

5. 選擇 Apply (套用)。

   組態修訂會在指定時間套用至代理程式。

編輯組態修訂

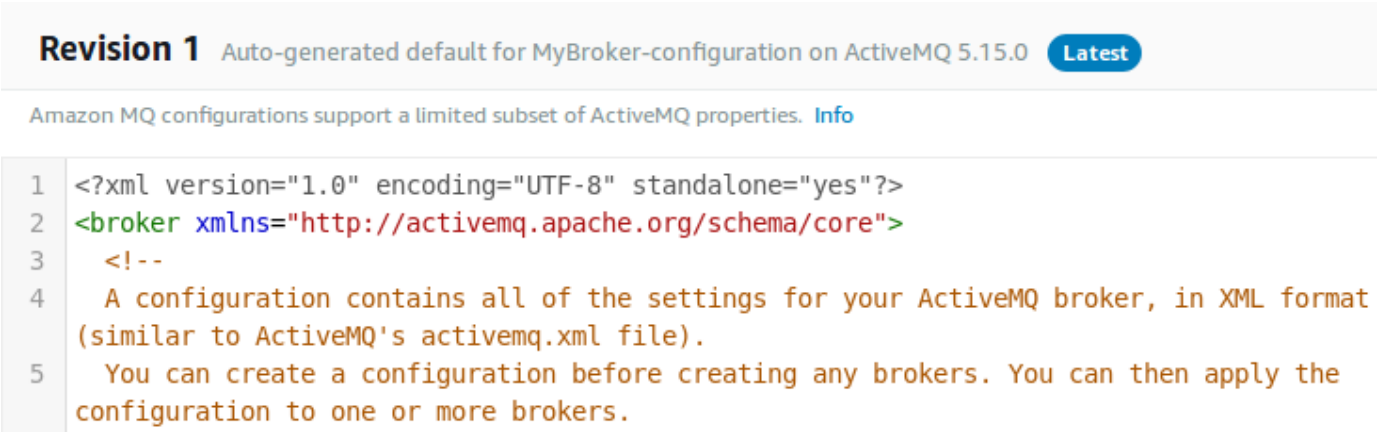
1. 登入 Amazon MQ 主控台。
2. 從代理程式清單，選取您的代理程式 (例如 MyBroker)，然後選擇 Edit (編輯)。
3. 在 ***MyBroker*** 頁面上，選取 Edit (編輯)。
4. 在 Edit ***MyBroker*** (編輯 MyBroker) 頁面的 Configuration (組態) 區段中，選取 Configuration (組態) 和 Revision (修訂)，然後選擇 Edit (編輯)。

> **Note**
>
> 除非您在建立代理程式時選取組態，否則一律會在 Amazon MQ 在建立代理程式時為您建立第一個組態修訂版。

在 ***MyBroker*** 頁面上，隨即顯示組態所使用的代理程式引擎類型和版本 (例如 Apache ActiveMQ 5.15.8)。

5. 在 Configuration details (組態詳細資訊) 標籤上，會顯示組態修訂編號、描述及 XML 格式的代理程式組態。

> **Note**
>
> 編輯目前的組態會建立新的組態版本。




6. 選擇 Edit configuration (編輯組態)，然後變更 XML 組態。
7. 選擇 Save (儲存)。

   Save revision (儲存修訂) 對話方塊隨即顯示。
8. (選用) 輸入 `A description of the changes in this revision`。
9. 選擇 Save (儲存)。

   隨即儲存組態的新修訂版。

> **Important**
>
> Amazon MQ 主控台會根據結構描述，自動清理無效及禁止的組態參數。如需詳細資訊和允許的完整 XML 參數清單，請參閱 Amazon MQ Broker Configuration Parameters。
> 對組態進行變更，不會立即將變更套用至代理程式。若要套用變更，您必須等待下一個維護時段或重新啟動代理程式。如需更多詳細資訊，請參閱 Amazon MQ 代理程式組態生命週期。
> 您目前無法刪除組態。

## Amazon MQ 組態中允許的元素

以下是 Amazon MQ 組態中允許之元素的詳細清單。如需詳細資訊，請參閱 Apache ActiveMQ 文件中的 XML 組態。

| 元素 |
| --- |
| abortSlowAckConsumerStrategy ( 屬性) |
| abortSlowConsumerStrategy ( 屬性) |
| authorizationEntry ( 屬性) |
| authorizationMap (子集合元素) |
| authorizationPlugin (子集合元素) |
| broker (屬性 \| 子集合元素) |
| cachedMessageGroupMapFactory ( 屬性) |
| compositeQueue (屬性 \| 子集合元素) |
| compositeTopic (屬性 \| 子集合元素) |
| constantPendingMessageLimitStrategy ( 屬性) |
| discarding ( 屬性) |

| 元素 |
| --- |
| discardingDLQBrokerPlugin （屬性） |
| fileCursor |
| fileDurableSubscriberCursor |
| fileQueueCursor |
| filteredDestination （屬性） |
| fixedCountSubscriptionRecoveryPolicy （屬性） |
| fixedSizedSubscriptionRecoveryPolicy （屬性） |
| forcePersistencyModeBrokerPlugin （屬性） |
| individualDeadLetterStrategy （屬性） |
| lastImageSubscriptionRecoveryPolicy |
| messageGroupHashBucketFactory （屬性） |
| mirroredQueue （屬性） |
| noSubscriptionRecoveryPolicy |
| oldestMessageEvictionStrategy （屬性） |
| oldestMessageWithLowestPriorityEvictionStrategy （屬性） |
| policyEntry （屬性 \| 子集合元素） |
| policyMap （子集合元素） |
| prefetchRatePendingMessageLimitStrategy （屬性） |
| priorityDispatchPolicy |
| priorityNetworkDispatchPolicy |

| 元素 |
| --- |
| queryBasedSubscriptionRecoveryPolicy ( 屬性) |
| queue( 屬性) |
| redeliveryPlugin (屬性 \| 子集合元素) |
| redeliveryPolicy ( 屬性) |
| redeliveryPolicyMap (子集合元素) |
| retainedMessageSubscriptionRecoveryPolicy (子集合元素) |
| roundRobinDispatchPolicy |
| sharedDeadLetterStrategy (屬性 \| 子集合元素) |
| simpleDispatchPolicy |
| simpleMessageGroupMapFactory |
| statisticsBrokerPlugin |
| storeCursor |
| storeDurableSubscriberCursor ( 屬性) |
| strictOrderDispatchPolicy |
| tempDestinationAuthorizationEntry ( 屬性) |
| tempQueue ( 屬性) |
| tempTopic ( 屬性) |
| timedSubscriptionRecoveryPolicy ( 屬性) |
| timeStampingBrokerPlugin ( 屬性) |
| topic( 屬性) |

| 元素 |
| --- |
| transportConnector （屬性） |
| uniquePropertyMessageEvictionStrategy （屬性） |
| virtualDestinationInterceptor （子集合元素） |
| virtualTopic （屬性） |
| vmCursor |
| vmDurableCursor |
| vmQueueCursor |

## Amazon MQ 組態中允許的元素及其屬性

以下是詳細清單，其中列出 Amazon MQ 組態中允許的元素及其屬性。如需詳細資訊，請參閱 Apache ActiveMQ 文件中的 XML 組態。

| 元素 | 屬性 |
| --- | --- |
| abortSlowAckConsumerStrategy | abortConnection |
| | checkPeriod |
| | ignoreIdleConsumers |
| | ignoreNetworkConsumers |
| | maxSlowCount |
| | maxSlowDuration |
| | maxTimeSinceLastAck |
| | name |
| abortSlowConsumerStrategy | abortConnection |

| 元素 | 屬性 |
| --- | --- |
| | checkPeriod |
| | ignoreNetworkConsumers |
| | maxSlowCount |
| | maxSlowDuration |
| | name |
| authorizationEntry | admin |
| | queue |
| | read |
| | tempQueue |
| | tempTopic |
| | topic |
| | write |
| broker | advisorySupport |
| | allowTempAutoCreationOnSend |
| | cacheTempDestinations |
| | consumerSystemUsagePortion |
| | dedicatedTaskRunner |
| | deleteAllMessagesOnStartup |
| | keepDurableSubsActive |
| | enableMessageExpirationOnActiveDurableSubs |

| 元素 | 屬性 |
| --- | --- |
| | maxPurgedDestinationsPerSweep |
| | maxSchedulerRepeatAllowed |
| | monitorConnectionSplits |
| | networkConnectorStartAsync |
| | offlineDurableSubscriberTaskSchedule |
| | offlineDurableSubscriberTimeout |
| | persistenceThreadPriority |
| | persistent |
| | populateJMSXUserID |
| | producerSystemUsagePortion |
| | rejectDurableConsumers |
| | rollbackOnlyOnAsyncException |
| | schedulePeriodForDestinationPurge |
| | schedulerSupport |
| | splitSystemUsageForProducersConsumers |
| | taskRunnerPriority |
| | timeBeforePurgeTempDestinations |
| | useAuthenticatedPrincipalForJMSXUserID |

| 元素 | 屬性 |
| --- | --- |
| | useMirroredQueues |
| | useTempMirroredQueues |
| | useVirtualDestSubs |
| | useVirtualDestSubsOnCreation |
| | useVirtualTopics |
| cachedMessageGroupMapFactory | cacheSize |
| compositeQueue | concurrentSend |
| | copyMessage |
| | forwardOnly |
| | name |
| | sendWhenNotMatched |
| compositeTopic | concurrentSend |
| | copyMessage |
| | forwardOnly |
| | name |
| | sendWhenNotMatched |
| conditionalNetworkBridgeFilterFactory | rateDuration |
| | rateLimit |
| | replayDelay |
| | replayWhenNoConsumers |

| 元素 | 屬性 |
| --- | --- |
| | selectorAware<br><br>支援於<br>Apache ActiveMQ 5.16.x |
| constantPendingMessageLimitStrategy | limit |
| discarding | deadLetterQueue |
| | enableAudit |
| | expiration |
| | maxAuditDepth |
| | maxProducersToAudit |
| | processExpired |
| | processNonPersistent |
| discardingDLQBrokerPlugin | dropAll |
| | dropOnly |
| | dropTemporaryQueues |
| | dropTemporaryTopics |
| | reportInterval |
| filteredDestination | queue |
| | selector |
| | topic |

| 元素 | 屬性 |
| --- | --- |
| fixedCountSubscriptionRecoveryPolicy | maximumSize |
| fixedSizedSubscriptionRecoveryPolicy | maximumSize |
| | useSharedBuffer |
| forcePersistencyModeBrokerPlugin | persistenceFlag |
| individualDeadLetterStrategy | destinationPerDurableSubscriber |
| | enableAudit |
| | expiration |
| | maxAuditDepth |
| | maxProducersToAudit |
| | processExpired |
| | processNonPersistent |
| | queuePrefix |
| | queueSuffix |
| | topicPrefix |
| | topicSuffix |
| | useQueueForQueueMessages |
| | useQueueForTopicMessages |
| messageGroupHashBucketFactory | bucketCount |
| | cacheSize |
| mirroredQueue | copyMessage |

| 元素 | 屬性 |
| --- | --- |
| | postfix |
| | prefix |
| oldestMessageEvictionStrategy | evictExpiredMessagesHighWat ermark |
| oldestMessageWithLowestPrio rityEvictionStrategy | evictExpiredMessagesHighWat ermark |
| policyEntry | advisoryForConsumed |
| | advisoryForDelivery |
| | advisoryForDiscardingMessages |
| | advisoryForFastProducers |
| | advisoryForSlowConsumers |
| | advisoryWhenFull |
| | allConsumersExclusiveByDefault |
| | alwaysRetroactive |
| | blockedProducerWarningInterval |
| | consumersBeforeDispatchStarts |
| | cursorMemoryHighWaterMark |
| | doOptimzeMessageStorage |
| | durableTopicPrefetch |
| | enableAudit |
| | expireMessagesPeriod |

| 元素 | 屬性 |
| --- | --- |
| | gcInactiveDestinations |
| | gcWithNetworkConsumers |
| | inactiveTimeoutBeforeGC |
| | inactiveTimoutBeforeGC |
| | includeBodyForAdvisory |
| | lazyDispatch |
| | maxAuditDepth |
| | maxBrowsePageSize |
| | maxDestinations |
| | maxExpirePageSize |
| | maxPageSize |
| | maxProducersToAudit |
| | maxQueueAuditDepth |
| | memoryLimit |
| | messageGroupMapFactoryType |
| | minimumMessageSize |
| | optimizedDispatch |
| | optimizeMessageStoreInFligh tLimit |
| | persistJMSRedelivered |
| | prioritizedMessages |

| 元素 | 屬性 |
| --- | --- |
| | producerFlowControl |
| | queue |
| | queueBrowserPrefetch |
| | queuePrefetch |
| | reduceMemoryFootprint |
| | sendAdvisoryIfNoConsumers |
| | sendFailIfNoSpace |
| | sendFailIfNoSpaceAfterTimeout<br>支援於<br>Apache ActiveMQ 5.16.4 及更新版本 |
| | sendDuplicateFromStoreToDLQ |
| | storeUsageHighWaterMark |
| | strictOrderDispatch |
| | tempQueue |
| | tempTopic |
| | timeBeforeDispatchStarts |
| | topic |
| | topicPrefetch |
| | useCache |
| | useConsumerPriority |

| 元素 | 屬性 |
| --- | --- |
| usePrefetchExtension | |
| prefetchRatePendingMessageLimitStrategy | multiplier |
| queryBasedSubscriptionRecoveryPolicy | query |
| queue | DLQ |
| | physicalName |
| redeliveryPlugin | fallbackToDeadLetter |
| | sendToDlqIfMaxRetriesExceeded |
| redeliveryPolicy | backOffMultiplier |
| | collisionAvoidancePercent |
| | initialRedeliveryDelay |
| | maximumRedeliveries |
| | maximumRedeliveryDelay |
| | preDispatchCheck |
| | queue |
| | redeliveryDelay |
| | tempQueue |
| | tempTopic |
| | topic |
| | useCollisionAvoidance |

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

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

| 元素 | 屬性 |
| --- | --- |
| | prefix |
| | selectorAware |
| | setOriginalDestination |
| | transactedSend |

Amazon MQ 父元素屬性

下列是父元素屬性的詳細說明。如需詳細資訊，請參閱 Apache ActiveMQ 文件中的 XML 組態。

主題

- 代理程式

代理程式

broker 是一種父系集合元素。

Attributes

networkConnectionStartAsync

為了減少網路的延遲，並且讓其他網路能夠及時開始，請使用 <networkConnectionStartAsync> 標籤。此標籤會指示代理程式，使用執行器來同時開始網路連線 (與代理程式的啟動非同步)。

預設：false

範例組態

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

## Amazon MQ 組態中允許的元素、子集合元素及其子元素

以下是詳細清單，其中列出 Amazon MQ 組態中允許的元素、子集合元素及其子元素。如需詳細資訊，請參閱 Apache ActiveMQ 文件中的 XML 組態。

| 元素 | 子集合元素 | 子元素 |
| --- | --- | --- |
| authorizationMap | authorizationEntries | authorizationEntry |
| | | tempDestinationAuthorizationEntry |
| | defaultEntry | authorizationEntry |
| | | tempDestinationAuthorizationEntry |
| | tempDestinationAuthorizationEntry | tempDestinationAuthorizationEntry |
| authorizationPlugin | map | authorizationMap |
| broker | destinationInterceptors | mirroredQueue |
| | | virtualDestinationInterceptor |
| | destinationPolicy | policyMap |
| | destinations | queue |
| | | tempQueue |
| | | tempTopic |
| | | topic |
| | networkConnectors | networkConnector |
| | persistenceAdapter | kahaDB |
| | plugins | authorizationPlugin |
| | | discardingDLQBrokerPlugin |

| 元素 | 子集合元素 | 子元素 |
| --- | --- | --- |
| | | forcePersistencyMo deBrokerPlugin |
| | | redeliveryPlugin |
| | | statisticsBrokerPl ugin |
| | | timeStampingBroker Plugin |
| | systemUsage | systemUsage |
| | transportConnector | name |
| | | updateClusterClients |
| | | rebalanceClusterCl ients |
| | | updateClusterClien tsOnRemove |
| compositeQueue | forwardTo | queue |
| | | tempQueue |
| | | tempTopic |
| | | topic |
| | | filteredDestination |
| compositeTopic | forwardTo | queue |
| | | tempQueue |
| | | tempTopic |

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

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

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

| 元素 | 子集合元素 | 子元素 |
| --- | --- | --- |
| redeliveryPlugin | redeliveryPolicyMap | redeliveryPolicyMap |
| redeliveryPolicyMap | defaultEntry | redeliveryPolicy |
| | redeliveryPolicyEntries | redeliveryPolicy |
| retainedMessageSubscriptionRecoveryPolicy | wrapped | fixedCountSubscriptionRecoveryPolicy |
| | | fixedSizedSubscriptionRecoveryPolicy |
| | | lastImageSubscriptionRecoveryPolicy |
| | | noSubscriptionRecoveryPolicy |
| | | queryBasedSubscriptionRecoveryPolicy |
| | | retainedMessageSubscriptionRecoveryPolicy |
| | | timedSubscriptionRecoveryPolicy |
| sharedDeadLetterStrategy | deadLetterQueue | queue |
| | | tempQueue |
| | | tempTopic |
| | | topic |
| virtualDestinationInterceptor | virtualDestinations | compositeQueue |

| 元素 | 子集合元素 | 子元素 |
| --- | --- | --- |
| | | compositeTopic |
| | | virtualTopic |

Amazon MQ 子元素屬性

以下是子元素屬性的詳細說明。如需詳細資訊，請參閱 Apache ActiveMQ 文件中的 XML 組態。

主題

- authorizationEntry
- networkConnector
- kahaDB
- systemUsage

authorizationEntry

authorizationEntry 是 authorizationEntries 子集合元素的子項。

屬性

管理|讀取|寫入

授予使用者群組的許可。如需更多詳細資訊，請參閱 一律設定授權映射。

如果您指定的授權映射不包含 activemq-webconsole 群組，您便無法使用 ActiveMQ Web 主控台，因為該群組未獲授權傳送或接收來自 Amazon MQ 代理程式的訊息。

預設：null

範例組態

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

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

networkConnector

networkConnector 是 networkConnectors 子集合元素的子項。

主題

- 屬性
- 範例組態

屬性

conduitSubscriptions

指定代理程式網路中的網路連線，是否將訂閱同一個目的地的多個使用者視為一個使用者。例如，如果 conduitSubscriptions 設定為 true，有兩個使用者連線到代理程式 B，並且從目的地使用，則代理程式 B 會透過網路連線，將這些訂閱合併為邏輯上對代理程式 A 的單一訂閱，因此只有一份訊息會從代理程式 A 轉傳到代理程式 B。

> **Note**
>
> 將 conduitSubscriptions 設定為 true 可減少重複的網路流量。不過，使用此屬性可能會影響到使用者之間的訊息負載平衡，而且可能會在某些情境中 (例如，使用 JMS 訊息選取器或持久的主題) 造成錯誤的行為。

預設：true

雙工

指定代理程式網路中的連線，是否用來產生和使用訊息。例如，如果代理程式 A 以非雙工模式建立至代理程式 B 的連線，則訊息只能從代理程式 A 轉傳到代理程式 B。不過，如果代理程式 A 建立至代理程式 B 的雙工連線，則代理程式 B 可以將訊息轉傳給代理程式 A，而不需設定 <networkConnector>。

預設：`false`

name

代理程式網路中橋接器的名稱。

預設：`bridge`

uri

在代理程式網路中，適用於兩個代理程式其中之一 (或多個代理程式) 的線路通訊協定端點。

預設：`null`

username

代理程式網路內代理程式共同的使用者名稱。

預設：`null`

範例組態

> **Note**
>
> 使用 `networkConnector` 來定義代理程式網路時，請不要包含代理程式共同使用者的密碼。

包含兩個代理程式的代理程式網路

使用此組態時，兩個代理程式會在代理程式網路中互連。網路連接器的名稱為 `connector_1_to_2`、代理程式共同的使用者名稱為 `myCommonUser`、連線為 `duplex`，而且 OpenWire 端點 URI 會加上 `static:` 前綴，表示代理程式之間的一對一連線。

```
<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>
```

如需更多詳細資訊，請參閱 Configure Network Connectors for Your Broker。

包含多個代理程式的代理程式網路

使用此組態時，多個代理程式會在代理程式網路中互連。網路連接器的名稱為 connector_1_to_2、代理程式共同的使用者名稱為 myCommonUser、連線為 duplex，而且 OpenWire 端點 URI 的逗點分隔清單，會加上 masterslave: 前綴，表示代理程式之間的容錯移轉連線。代理程式之間的容錯移轉並非隨機進行，而且會無限期地持續重新嘗試連線。

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

> **Note**
>
> 我們建議針對代理程式網路使用 masterslave: 前綴。此前綴與更明確的 static:failover:()?randomize=false&maxReconnectAttempts=0 語法相同。

> **Note**
>
> 此 XML 組態不允許使用空格。

kahaDB

kahaDB 是 persistenceAdapter 子集合元素的子項。

屬性

concurrentStoreAndDispatchQueues

指定是否針對佇列使用並行儲存與調配。如需更多詳細資訊，請參閱 對於具有緩慢消費者的佇列，停用並行存放和分派。

預設：true

cleanupOnStop

> 支援於
>
> Apache ActiveMQ 15.16.x 及更新版本

若已停用，廢棄項目收集和清理就不會發生在代理程式停止時，這會加快關機程序。在大型資料庫或排程器資料庫的情況下，提高速度非常有用。

預設：`true`

journalDiskSyncInterval

若為 `journalDiskSyncStrategy=periodic`，要在何時執行磁碟同步的間隔 (毫秒)。如需詳細資訊，請參閱 Apache ActiveMQ kahaDB 文件。

預設：`1000`

journalDiskSyncStrategy

> 支援於
>
> Apache ActiveMQ 15.14.x 及更高版本

設定磁碟同步政策。如需詳細資訊，請參閱 Apache ActiveMQ kahaDB 文件。

預設：`always`

> Note
>
> ActiveMQ 文件會陳述資料遺失限制為 `journalDiskSyncInterval` 的持續時間，其預設值為 1 秒。資料遺失的時間長度可大於此間隔，但很難保持精確。請謹慎使用。

preallocationStrategy

設定代理程式將如何嘗試在需要新日誌檔案時，預先配置日誌檔案。如需詳細資訊，請參閱 Apache ActiveMQ kahaDB 文件。

預設：sparse_file

範例組態

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 子集合元素的子項。它可控制代理程式在減慢生產者速度前將使用的空間量上限。如需詳細資訊，請參閱 Apache ActiveMQ 文件中的生產者流程控制。

子元素

memoryUsage

memoryUsage 是 systemUsage 子元素的子項。它可管理記憶體用量。使用 memoryUsage 追蹤某個項目的使用量，以便發揮生產力來控制工作集用量。如需詳細資訊，請參閱 Apache ActiveMQ 中的結構描述。

子元素

memoryUsage 是 memoryUsage 子元素的子項。

屬性

percentOfJvmHeap

介於 0 (包含在內) 到 70 (不包含在內) 之間的整數。

預設：70

屬性

sendFailIfNoSpace

設定在沒有可用空間的情況下 send() 方法是否應失敗。預設值為 false，這會封鎖 send() 方法，直到有空間可用為止。如需詳細資訊，請參閱 Apache Active MQ 中的結構描述。

預設：`false`

sendFailIfNoSpaceAfterTimeout

預設：`null`

範例組態

Example

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

# 管理 Amazon MQ for ActiveMQ 引擎版本

Apache ActiveMQ 會根據語義版本控制規格將版本號碼組織為 `X.Y.Z`。在 Amazon MQ for ActiveMQ 實作中，`X.Y` 表示主要版本，而 `Z` 代表次要版本號碼。如果主要版本號碼發生變更，Amazon MQ 會將版本變更視為主要版本變更。例如，從 5.15 版升級至 5.16 版視為主要版本升級。如果只有次要版本號碼有所改變，則視為次要版本變更。例如，從 5.15.14 版升級至 5.15.15 版被視為次要版本升級。

Amazon MQ for ActiveMQ 目前支援下列引擎版本的 Apache ActiveMQ。

| 主要版本 | 次要版本 |
| --- | --- |
| ActiveMQ 5.17 | • 5.17.6 (建議使用) |
| ActiveMQ 5.16 | • 5.16.7 |
| ActiveMQ 5.15 | • 5.15.16 |

下列次要版本已棄用。

已棄用的次要版本

| 主要版本 | 次要版本 |
| --- | --- |
| ActiveMQ 5.17 | • 5.17.3<br>• 5.17.2<br>• 5.17.1 |
| ActiveMQ 5.16 | • 5.16.5<br>• 5.16.4<br>• 5.16.3<br>• 5.16.2 |
| ActiveMQ 5.15 | • 5.15.15<br>• 5.15.14<br>• 5.15.13<br>• 5.15.12<br>• 5.15.10<br>• 5.15.9<br>• 5.15.8<br>• 5.15.6<br>• 5.15.0 |

當您建立新的 Amazon MQ for ActiveMQ 代理程式時，您可以指定任何支援的 ActiveMQ 引擎版本。如果您使用 AWS Management Console 來建立代理程式，Amazon MQ 會自動預設為最新的引擎版本號碼。如果您使用 AWS CLI 或 Amazon MQ API 來建立代理程式，則需要引擎版本號碼。如果您不提供版本號碼，則操作會導致例外狀況。如需進一步了解，請參閱 AWS CLI 命令參考中的 `create-broker` 和 Amazon MQ REST API 參考中的 `CreateBroker`。

主題

- 主要和次要版本升級
- 列出支援的引擎版本

### 主要和次要版本升級

透過 Amazon MQ，即可控制將代理程式升級至新版本的時機。若已啟用自動次要版本升級，Amazon MQ 將自動將您的代理引擎升級到由 Amazon MQ 發行並支援的新次要版本。

若要執行主要版本升級，您必須手動升級代理程式的引擎版本號碼。次要和主要版本升級可以在您排定的維護時段內，與其他代理程式修補作業同時發生。若您選擇不要自動次要版本升級，可以按照與主要版本相同的程序，將代理程式手動升級至新支援的次要版本。

如需更新代理程式偏好設定以啟用或停用次要版本升級，以及手動升級代理程式的詳細資訊，請參閱 the section called “升級引擎版本”。

### 列出支援的引擎版本

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

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

若要依照引擎和執行個體類型篩選結果，請使用 `--engine-type` 和 `--host-instance-type` 選項，如下所示。

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

例如，若要篩選 ActiveMQ 和 `mq.m5.large` 執行個體類型的結果，請將 *`engine-type`* 取代為 `ACTIVEMQ` 以及將 *`instance-type`* 取代為 `mq.m5.large`。

## 搭配使用 Java Message Service (JMS) 與 ActiveMQ 的運作範例

以下範例顯示如何以程式設計方式使用 ActiveMQ：

- OpenWire 範例 Java 程式碼連接至代理程式、建立佇列，並傳送及接收訊息。如需細節和詳細說明，請參閱 Connecting a Java application to your broker。
- MQTT 範例 Java 程式碼會連接至代理程式、建立主題，以及發佈和接收訊息。
- STOMP+WSS 範例 Java 程式碼會連接至代理程式、建立主題，以及發佈和接收訊息。

## 先決條件

### 啟用 VPC 屬性

若要確保代理程式可以在 VPC 內存取，您必須啟用 `enableDnsHostnames` 和 `enableDnsSupport` VPC 屬性。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的 VPC 中的 DNS Support。

### 啟用傳入連線

1. 登入 Amazon MQ 主控台。
2. 從代理程式清單中，選擇您的代理程式名稱 (例如，MyBroker)。
3. 在 ***MyBroker*** 頁面的 Connections (連線) 區段中，記下代理程式 Web 主控台 URL 和線路通訊協定的位址和連接埠。
4. 在 Details (詳細資訊) 區段的 Security and network (安全與網路) 下，選擇您的安全群組名稱或 。

   隨即會顯示 EC2 儀表板的 Security Groups (安全群組) 頁面。
5. 從安全群組清單選擇您的安全群組。
6. 在頁面的最下方，選擇 Inbound (傳入)，然後選擇 Edit (編輯)。
7. 在 Edit inbound rules (編輯傳入規則) 對話方塊中，為您要公開存取的每個 URL 或端點新增規則 (下列範例顯示如何針對代理程式 Web 主控台執行此動作)。

   a. 選擇 Add Rule (新增規則)。

   b. 針對 Type (類型)，選擇 Custom TCP (自訂 TCP)。

   c. 針對 Port Range (連接埠範圍)，輸入 Web 主控台連接埠 (8162)。

   d. 針對 Source (來源)，讓 Custom (自訂) 保持已選取狀態，然後輸入您希望能夠存取 Web 主控台的系統 IP 地址 (例如，`192.0.2.1`)。

   e. 選擇 Save (儲存)。

      您的代理程式現在已可接受傳入連線。

### 新增 Java 相依性

#### OpenWire

將 `activemq-client.jar` 和 `activemq-pool.jar` 套件新增到您的 Java 類別路徑。以下範例顯示 Maven 專案 `pom.xml` 檔案中的此等依存關係。

```
<dependencies>
    <dependency>
        <groupId>org.apache.activemq</groupId>
        <artifactId>activemq-client</artifactId>
        <version>5.15.8</version>
    </dependency>
    <dependency>
        <groupId>org.apache.activemq</groupId>
        <artifactId>activemq-pool</artifactId>
        <version>5.15.8</version>
    </dependency>
</dependencies>
```

如需 `activemq-client.jar` 的詳細資訊，請參閱 Apache ActiveMQ 文件中的初始組態。

MQTT

將 `org.eclipse.paho.client.mqttv3.jar` 套件新增至您的 Java 類別路徑。以下範例顯示 Maven 專案 `pom.xml` 檔案中的此一依存關係。

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

如需 `org.eclipse.paho.client.mqttv3.jar` 的詳細資訊，請參閱 Eclipse Paho Java Client。

STOMP+WSS

將以下套件新增到您的 Java 類別路徑：

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

- `jackson-databind.jar`

以下範例顯示 Maven 專案 `pom.xml` 檔案中的此等依存關係。

```
<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>
```

如需詳細資訊，請參閱 Spring Framework 文件中的 STOMP Support。

## AmazonMQExample.java

> **Important**
>
> 在以下的範例程式碼中，生產者和消費者會在單一執行緒中執行。對於生產系統 (或測試代理程式執行個體容錯移轉)，請確定您的生產者和消費者在不同的主機或執行緒上執行。

### OpenWire

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

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

    import javax.jms.*;

    public class AmazonMQExample {

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

    public static void main(String[] args) throws JMSException {
        final ActiveMQConnectionFactory connectionFactory =
```

```
                createActiveMQConnectionFactory();
        final PooledConnectionFactory pooledConnectionFactory =
                createPooledConnectionFactory(connectionFactory);

        sendMessage(pooledConnectionFactory);
        receiveMessage(connectionFactory);

        pooledConnectionFactory.stop();
    }

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

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

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

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

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

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

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

```
    }

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

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

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

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

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

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

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

    private static PooledConnectionFactory
    createPooledConnectionFactory(ActiveMQConnectionFactory connectionFactory) {
        // Create a pooled connection factory.
        final PooledConnectionFactory pooledConnectionFactory =
                new PooledConnectionFactory();
        pooledConnectionFactory.setConnectionFactory(connectionFactory);
        pooledConnectionFactory.setMaxConnections(10);
        return pooledConnectionFactory;
    }
```

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

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

### MQTT

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

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

    public class AmazonMQExampleMqtt implements MqttCallback {

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

    public static void main(String[] args) throws Exception {
```

```
        new AmazonMQExampleMqtt().run();
    }

    private void run() throws MqttException, InterruptedException {

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

        // Create the MQTT client and specify the connection options.
        final String clientId = "abc123";
        final MqttClient client = new MqttClient(WIRE_LEVEL_ENDPOINT, clientId);
        final MqttConnectOptions connOpts = new MqttConnectOptions();

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

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

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

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

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

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

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

    @Override
```

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

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

### STOMP+WSS

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

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

    import java.lang.reflect.Type;

    public class AmazonMQExampleStompWss {

    // Specify the connection parameters.
    private final static String DESTINATION = "/queue";
    private final static String WIRE_LEVEL_ENDPOINT =
```

```
            "wss://b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-
east-2.amazonaws.com:61619";
    private final static String ACTIVE_MQ_USERNAME = "MyUsername123";
    private final static String ACTIVE_MQ_PASSWORD = "MyPassword456";

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

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

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

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

        final WebSocketHttpHeaders headers = new WebSocketHttpHeaders();

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

        final StompSessionHandler sessionHandler = new MySessionHandler();

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

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

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

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

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

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

# ActiveMQ 教學

以下教學說明如何建立及連線至 ActiveMQ 代理程式。若要使用 ActiveMQ Java 範例程式碼，您必須安裝 Java 標準版開發套件，並對程式碼進行一些變更。

主題

- 建立和設定 ActiveMQ 代理程式
- 建立和設定 Amazon MQ 代理程式網路
- 將 Java 應用程式連線到您的 Amazon MQ 代理程式
- 整合 ActiveMQ 代理程式與 LDAP
- 建立和管理 ActiveMQ 代理程式使用者

## 建立和設定 ActiveMQ 代理程式

代理程式是在 Amazon MQ 上執行的訊息代理程式環境。這是 Amazon MQ 的基本建置區塊。代理程式執行個體類別 (`m5`、`t3`) 和大小 (`large`、`micro`) 的合併說明是代理程式執行個體類型 (例如，`mq.m5.large`)。如需更多詳細資訊，請參閱 中介裝置。

第一個最常見的 Amazon MQ 任務是建立代理程式。以下範例顯示如何使用 AWS Management Console 來建立和設定代理程式。

主題



## 步驟 1：設定基本代理程式設定

1. 登入 Amazon MQ 主控台。
2. 在 Select broker engine (選取代理程式引擎) 頁面上，選擇 Apache ActiveMQ。
3. 在 Select deployment and storage (選取部署和儲存) 頁面的 Deployment mode and storage type (部署模式和儲存類型) 區段中，執行下列動作：

   a. 選擇 Deployment mode (部署模式) (例如，作用中/待命代理程式)。如需詳細資訊，請參閱 Broker Architecture。

      - 單一執行個體代理程式是由一個可用區域中的一個代理程式組成。代理程式會與您的應用程式以及 Amazon EBS 或 Amazon EFS 儲存磁碟區進行通訊。如需更多詳細資訊，請參閱 Amazon MQ 單一執行個體代理程式。
      - 高可用性的作用中/待命代理程式是由兩個不同可用區域中的兩個代理程式所組成，並設定於備援組合中。這些代理程式會與您的應用程式及 Amazon EFS 同步通訊。如需更多詳細資訊，請參閱 高可用性的 Amazon MQ 作用中/待命代理程式。
      - 如需代理程式網路的範例藍圖詳細資訊，請參閱 藍圖範例。

   b. 選擇 Storage type (儲存體類型) (例如，EBS)。如需更多詳細資訊，請參閱 Storage。

      > **Note**
      >
      > Amazon EBS 會複寫單一可用區域內的資料，且不支援 ActiveMQ 作用中/待命部署模式。

   c. 選擇 Next (下一步)。
4. 在 Configure settings (進行設定) 頁面的 Details (詳細資訊) 區段中，執行以下動作：

a. 輸入 Broker name (代理程式名稱)。

> **Important**
>
> 請勿在代理程式名稱中加入個人身分識別資訊 (PII) 或其他機密或敏感資訊。包括 CloudWatch Logs 在內的其他 AWS 服務可存取代理程式名稱。代理程式名稱不適用於私有或敏感資料。

b. 選擇 Broker instance type (代理程式執行個體類型) (例如，mq.m5.large)。如需更多詳細資訊，請參閱 Broker instance types。

5. 在 ActiveMQ Web Console access (ActiveMQ Web 主控台存取) 區段中，提供 Username (使用者名稱) 和 Password (密碼)。以下限制適用於代理程式使用者名稱和密碼：

- 使用者名稱只能包含英數字元、破折號、句點、底線和波狀符號 (- . _ ~)。
- 密碼必須至少有 12 個字元、包含至少 4 個唯一字元，而且不得包含逗號、冒號或等號 (,: =)。

> **Important**
>
> 請勿在代理程式使用者名稱中加入個人身分識別資訊 (PII) 或其他機密或敏感資訊。包括 CloudWatch Logs 在內的其他 AWS 服務可存取代理程式使用者名稱。代理程式使用者名稱不適用於私有或敏感資料。

## 步驟 2：(選用) 設定其他代理程式設定

> **Important**
>
> - 子網路 – 單一執行個體代理程式需要一個子網路 (例如，預設子網路)。作用中/待命代理程式需要兩個子網路。
> - 安全群組 – 單一執行個體代理程式和作用中/待命代理程式兩者都至少需要一個安全群組 (例如，預設安全群組)。
> - VPC – 代理程式的子網路和安全群組必須位於同一個 VPC 中。不支援 EC2-Classic 資源。Amazon MQ 僅支援預設 VPC 租用，而不支援專用的 VPC 租用。
> - 加密 – 選擇客戶主金鑰來加密您的資料。請參閱 靜態加密。

- 公開存取性 – 停用公開存取性會使得代理程式只能在 VPC 內存取。如需詳細資訊，請參閱偏好無法公開存取的代理程式 及 在沒有公開存取性的情況下存取代理程式 Web 主控台。

1. 展開 Additional settings (其他設定) 區段。
2. 在 Configuration (組態) 區段中，選擇 Create a new configuration with default values (使用預設值建立新組態) 或 Select an existing configuration (選取現有的組態)。如需詳細資訊，請參閱 組態 及 Amazon MQ Broker Configuration Parameters。
3. 在 Logs (日誌) 區段中，選擇要將 General (一般) 日誌還是 Audit (稽核) 日誌發佈至 Amazon CloudWatch Logs 。如需更多詳細資訊，請參閱 Configuring Amazon MQ to publish logs to Amazon CloudWatch Logs。

   **Important**

   在使用者建立或重新啟動代理程式之前，如果您未將 `CreateLogGroup` 許可新增至 Amazon MQ 使用者，則 Amazon MQ 不會建立日誌群組。
   如果您未對 Amazon MQ 設定資源型政策，則代理程式無法將日誌發佈到 CloudWatch Logs。

4. 在 Network and security (網路與安全) 區段中，設定代理程式的連線：

   a. 執行下列任意一項：

      - 選擇 Use the default VPC, subnet(s), and security group(s) (使用預設的 VPC、子網路和安全群組)。
      - 選擇 Select existing VPC, subnet(s), and security group(s) (選取現有的 VPC、子網路和安全群組)。
        1. 如果您選擇此選項，即可在 Amazon VPC 主控台上建立新的 Virtual Private Cloud (VPC)，選取現有的 VPC，或選取預設 VPC。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的什麼是 Amazon VPC？。
        2. 建立或選取 VPC 後，您可以在 Amazon VPC 主控台上建立新的 Subnet(s) (子網路)，或選取現有的子網路。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的 VPC 和子網路。
        3. 在建立或選取子網路後，您可以選取 Security group(s) (安全群組)。

   b. 選擇將用於加密您資料的客戶主金鑰 (CMK)。請參閱 靜態加密。

   c. 選擇代理程式的 Public accessibility (公開存取性)。

5. 在 Maintenance (維護) 區段中，設定代理程式的維護排程：

   a. 若要在 Apache 發佈新版本時，將代理程式升級至該版本，請選擇 Enable automatic minor version upgrades (啟用自動次要版本升級)。自動升級會發生於由星期幾、一天中的時間 (24 小時制) 和時區 (預設為 UTC) 所定義的維護時段期間。

   > **Note**
   >
   > 對於作用中/待命代理程式，如果其中一個代理程式執行個體進行維護，Amazon MQ 需要一段時間才能將非作用中執行個體停止服務。這可讓狀況良好的待命執行個體變成作用中，並開始接受傳入的通訊。

   b. 執行下列任意一項：

   - 若要允許 Amazon MQ 自動選取維護時段，請選擇 No preference (無偏好設定)。
   - 若要設定自訂的維護時段，請選擇 Select maintenance window (選取維護時段)，然後指定升級的 Start day (開始日) 和 Start time (開始時間)。

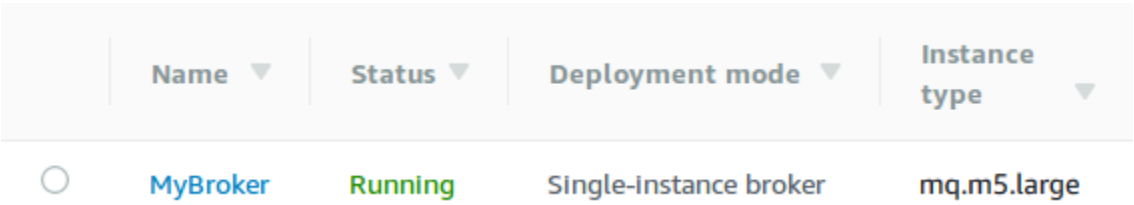
## 步驟 3：完成代理程式的建立

1. 選擇 部署 。

   當 Amazon MQ 建立您的代理程式時，其會顯示 Creation in progress (正在建立) 狀態。

   建立代理程式大約需要 15 分鐘。

   成功建立代理程式後，Amazon MQ 會顯示 Running (執行中) 狀態。




2. 選擇 ***MyBroker***。

   在 ***MyBroker*** 頁面的 Connect (連線) 區段中，請記下代理程式的 ActiveMQ web console (ActiveMQ Web 主控台) URL，例如：

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

此外，請記下代理程式的線路通訊協定端點。以下是 OpenWire 端點的範例：

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

> **Note**
>
> 對於作用中/待命代理程式，Amazon MQ 會提供兩個 ActiveMQ Web 主控台 URL，但一次只有一個作用中的 URL。同樣地，Amazon MQ 為每個線路通訊協定提供兩個端點，但每個配對中一次只有一個作用中的端點。-1 和 -2 尾碼表示備援組合。如需詳細資訊，請參閱 Broker Architecture。
>
> 對於線路通訊協定端點，您可以允許應用程式使用容錯移轉傳輸連線到任一端點。

## 編輯代理程式引擎版本、執行個體類型、CloudWatch Logs，以及維護偏好設定

除了編輯代理程式組態和管理組態修訂之外，您還可以設定代理程式特有的偏好設定。

> **Note**
>
> 所有偏好設定 (但自動次要版本升級的偏好設定除外) 需要您排定修改時間。如需更多詳細資訊，請參閱 Amazon MQ 代理程式組態生命週期。

以下範例示範如何使用 AWS Management Console 來編輯 Amazon MQ ActiveMQ 代理程式偏好設定。

編輯 ActiveMQ 代理程式選項

1. 登入 Amazon MQ 主控台。
2. 從代理程式清單，選取您的代理程式 (例如 MyBroker)，然後選擇 Edit (編輯)。
3. 在 Edit ***MyBroker*** (編輯 MyBroker) 頁面上，於 Specifications (規格) 區段中，選取 Broker engine version (代理程式引擎版本) 或 Broker Instance type (代理程式執行個體類型)。
4. 在 Configuration (組態) 區段中，選取代理程式的組態和修訂。如需更多詳細資訊，請參閱 Creating and applying broker configurations。
5. 在 Security and network (安全與網路) 區段中，從 Security group(s) (安全群組) 下拉式清單選取群組，或選擇 Create a new security group (建立新的安全群組) 以開啟 Amazon VPC 主控台。

6. 在 CloudWatch Logs 區段中，選擇要將 General (一般) 日誌還是 Audit (稽核) 日誌發佈至 Amazon CloudWatch Logs 。

   如需設定 ActiveMQ 代理程式的 CloudWatch Logs 的詳細資訊，請參閱 Configuring Amazon MQ to publish logs to Amazon CloudWatch Logs。

   > **Important**
   >
   > 在使用者建立或重新啟動代理程式之前，如果您未將 `CreateLogGroup` 許可新增至 Amazon MQ 使用者，則 Amazon MQ 不會建立日誌群組。
   > 如果您未對 Amazon MQ 設定資源型政策，則代理程式無法將日誌發佈到 CloudWatch Logs。

7. 在 Maintenance (維護) 區段中，設定代理程式的維護排程：

   若要在 AWS 發佈新版本時，將代理程式升級至該版本，請選擇 Enable automatic minor version upgrades (啟用自動次要版本升級)。自動升級會發生於由星期幾、一天中的時間 (24 小時制) 和時區 (預設為 UTC) 所定義的維護時段期間。

   > **Note**
   >
   > 對於作用中/待命代理程式，如果其中一個代理程式執行個體進行維護，Amazon MQ 需要一段時間才能將非作用中執行個體停止服務。這可讓狀況良好的待命執行個體變成作用中，並開始接受傳入的通訊。

8. 選擇 Schedule Modifications (排程修改)。

   > **Note**
   >
   > 如果您只選擇 Enable automatic minor version upgrades (啟用自動次要版本升級)，按鈕會變更為 Save (儲存)，因為無需重新啟動代理程式。

   會在指定時間將您的偏好設定套用至代理程式。

# 建立和設定 Amazon MQ 代理程式網路

代理程式網路由多個同時作用中單一執行個體代理程式或作用中/待命代理程式組成。您可以在各種拓撲 (例如，集中器、中樞和支點、樹狀結構或網格) 中設定代理程式網路，這取決於您的應用程式的需求，例如高可用性和可擴展性。例如，中樞和支點代理程式網路網路可以提高恢復能力，如果一個代理程式無法連線，則會保留訊息。具有集中器拓撲的代理程式網路可以從接受傳入訊息的大量代理程式收集訊息，並將它們集中到更多中央代理程式，以便更妥善處理許多傳入訊息的載入。在此教學中，您將了解如何使用來源與目的拓撲來建立包含兩個代理程式的代理程式網路。

如需概念性的概觀和詳細的組態資訊，請參閱下列內容：

- Amazon MQ 代理程式網路
- 正確地設定您的代理程式網路
- `networkConnector`
- `networkConnectionStartAsync`
- ActiveMQ 文件中的 Networks of Brokers (代理程式網路)

您可以使用 Amazon MQ 主控台來建立 Amazon MQ 代理程式網路。由於您可以同時開始建立兩個代理程式，因此這項程序約需 15 分鐘的時間完成。

主題



## 先決條件

若要建立代理程式網路，您必須具有下列項目：

- 兩個或多個同時運作中的代理程式 (在本教學課程中命名為 `MyBroker1` 和 `MyBroker2`)。如需關於建立代理程式的詳細資訊，請參閱Creating and configuring a broker。
- 這兩個代理程式必須位於同一個 VPC 或對等 VPC 中。如需 VPC 的詳細資訊，請參閱《Amazon VPC 使用者指南》中的什麼是 Amazon VPC？和《Amazon VPC 對等互連指南》中的什麼是 VPC 對等互連？。

> **Important**
>
> 如果您沒有預設的 VPC、子網路或安全群組，則必須先建立這些項目。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的下列項目：
>
> - 建立預設的 VPC
> - 建立預設子網路
> - 建立安全群組

- 兩個使用者對於兩個代理程式具有相同的登入憑證。如需有關建立使用者的詳細資訊，請參閱建立和管理 ActiveMQ 代理程式使用者。

> **Note**
>
> 將 LDAP 身分驗證與代理程式網路整合時，請確定使用者以 ActiveMQ 代理程式及 LDAP 使用者存在。

下列的範例使用兩個單一執行個體代理程式。不過，您可以使用運作中/待命的代理程式或代理程式部署模式的組合，來建立代理程式的網路。

### 步驟 1：允許代理程式之間的流量

建立代理程式之後，您必須允許裝置之間的流量。

1. 在 Amazon MQ 主控台中，於 MyBroker2 頁面的 Details (詳細資訊) 區段中, 在 Security and network (安全與網路) 之下，選擇您的安全群組名稱或 。

   隨即會顯示 EC2 儀表板的 Security Groups (安全群組) 頁面。
2. 從安全群組清單選擇您的安全群組。
3. 在頁面的最下方，選擇 Inbound (傳入)，然後選擇 Edit (編輯)。
4. 在 Edit inbound rules (編輯傳入規則) 對話方塊中，新增 OpenWire 端點的規則。
   a. 選擇 Add Rule (新增規則)。
   b. 針對 Type (類型)，選擇 Custom TCP (自訂 TCP)。
   c. 針對 Port Range (連接埠範圍)，輸入 OpenWire 連接埠 (61617)。

d. 執行下列任意一項：

- 如果您想限制對特定 IP 地址的存取，則請讓 Source (來源) 維持選取 Custom (自訂)，然後輸入 `MyBroker1` 的 IP 地址，後面接 `/32`。(這會將 IP 地址轉換為有效的 CIDR 記錄)。如需詳細資訊，請參閱彈性網路界面。

> **Tip**
>
> 若要擷取 `MyBroker1` 的 IP 地址，請在 Amazon MQ 主控台上，選擇代理程式的名稱，然後瀏覽至 Details (詳細資訊) 區塊。

- 如果全部代理程式都為私有，並隸屬於同一個 VPC，則請讓 Source (來源) 維持選取 Custom (自訂)，然後輸入您所編輯安全群組的 ID。

> **Note**
>
> 用於公有代理程式時，您必須使用 IP 地址來限制存取。

e. 選擇 Save (儲存)。

您的代理程式現在已可接受傳入連線。

## 步驟 2：設定代理程式的網路連接器

在允許代理程式之間的流量之後，您必須針對其中一個代理程式，設定其網路連接器。

1. 編輯代理程式 `MyBroker1` 的組態版本。

a. 在 MyBroker1 (MyBroker1) 頁面上，選擇 Edit (編輯)。

b. 在 Edit MyBroker1 (編輯 MyBroker1) 頁面的 Configuration (組態) 區塊中，選擇 View (檢視)。

隨即會顯示組態所使用的代理程式引擎類型和版本 (例如 Apache ActiveMQ 5.15.0 (Apache ActiveMQ 5.15.0))。

c. 在 Configuration details (組態詳細資訊) 標籤上，會顯示組態修訂編號、描述及 XML 格式的代理程式組態。

d. 選擇 Edit Configuration (編輯組態)。

e. 在組態檔案的底部，將 `<networkConnectors>` 部分改成非註解，然後加入下列的資訊：

- 網路連接器的 name。
- 兩個代理程式共同的 ActiveMQ Web 主控台username。
- 啟用 duplex 連線。
- 執行下列任意一項：
  - 如果您要將代理程式連接到單一執行個體的代理程式，請針對 static: 使用 uri 前綴和 OpenWire 端點 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>
```

  - 如果您要將代理程式連接到運作中/待命的代理程式，請以下列查詢參數 ?randomize=false&maxReconnectAttempts=0 針對兩個代理程式使用 static +failover 傳輸和 OpenWire 端點 uri。例如：

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

> **Note**
>
> 請勿納入 ActiveMQ 使用者的登入憑證。

f. 選擇 Save (儲存)。

g. 在 Save revision (儲存修改) 對話方塊中，輸入 Add network of brokers connector for MyBroker2。

h. 選擇 Save (儲存) 以儲存組態的新版本。

2. 編輯 MyBroker1 以設定立即套用最新的組態修改內容。

a. 在 MyBroker1 (MyBroker1) 頁面上，選擇 Edit (編輯)。
b. 在 Edit MyBroker1 (編輯 MyBroker1) 頁面的 Configuration (組態) 區段中，選擇 Schedule Modifications (排程修改)。
c. 在 Schedule broker modifications (排程代理程式修改) 區塊中，選擇 Immediately (立即) 套用修改。
d. 選擇 Apply (套用)。

   MyBroker1 會重新啟動並套用您組態的修改內容。

代理程式網路已建立。

## 後續步驟

設定代理程式網路之後，您可以藉由產生和使用訊息，來測試該網路。

> **Important**
>
> 請務必針對MyBroker1連接埠 8162 (適用於 ActiveMQ Web 主控台) 和連接埠 61617 (適用於 OpenWire 端點) 上的代理程式，從您的本機電腦啟用傳入連線。
> 您可能也需要調整安全群組的設定，來允許生產者和使用者連線到代理程式網路。

1. 在 Amazon MQ 主控台中，瀏覽至 Connections (連線) 區塊，然後記下代理程式 MyBroker1 的 ActiveMQ Web 主控台端點。
2. 瀏覽至代理程式 MyBroker1 的 ActiveMQ Web 主控台。
3. 若要確認網路橋接器已連接，請選擇 Network (網路)。

   在 Network Bridges (網路橋接器) 區段中，MyBroker2 的名稱和地址會列於 Remote Broker (遠端代理程式) 和 Remote Address (遠端地址) 欄中。
4. 從對代理程式 MyBroker2 具有存取權限的任何機器，來建立使用者。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
```

使用者可連線到 MyBroker2 的 OpenWire 端點，並開始使用來自佇列 MyQueue 的訊息。

5. 從對代理程式 MyBroker1 具有存取權限的任何機器，來建立生產者和傳送一些訊息。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
```

生產者可連線到 MyBroker1 的 OpenWire 端點，並開始生產要傳送到佇列 MyQueue 的持久性訊息。

## 將 Java 應用程式連線到您的 Amazon MQ 代理程式

建立 Amazon MQ ActiveMQ 代理程式後，您可以將應用程式連接到它。下列範例示範如何使用 Java 訊息服務 (JMS) 建立與代理程式的連線、建立佇列以及傳送訊息。如需完整的作用中 Java 範例，請參閱 Working Java Example。

您可以使用多種 ActiveMQ 用戶端連線至 ActiveMQ 代理程式。建議使用 ActiveMQ 用戶端。

主題

- 先決條件
- 建立訊息生產者和傳送訊息
- 建立訊息消費者和接收訊息

### 先決條件

#### 啟用 VPC 屬性

若要確保代理程式可以在 VPC 內存取，您必須啟用 enableDnsHostnames 和 enableDnsSupport VPC 屬性。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的 VPC 中的 DNS Support。

#### 啟用傳入連線

1. 登入 Amazon MQ 主控台。

2. 從代理程式清單中，選擇您的代理程式名稱 (例如，MyBroker)。
3. 在 ***MyBroker*** 頁面的 Connections (連線) 區段中，記下代理程式 Web 主控台 URL 和線路通訊協定的位址和連接埠。
4. 在 Details (詳細資訊) 區段的 Security and network (安全與網路) 下，選擇您的安全群組名稱或 。

   隨即會顯示 EC2 儀表板的 Security Groups (安全群組) 頁面。
5. 從安全群組清單選擇您的安全群組。
6. 在頁面的最下方，選擇 Inbound (傳入)，然後選擇 Edit (編輯)。
7. 在 Edit inbound rules (編輯傳入規則) 對話方塊中，為您要公開存取的每個 URL 或端點新增規則 (下列範例顯示如何針對代理程式 Web 主控台執行此動作)。

   a. 選擇 Add Rule (新增規則)。

   b. 針對 Type (類型)，選擇 Custom TCP (自訂 TCP)。

   c. 針對 Port Range (連接埠範圍)，輸入 Web 主控台連接埠 (8162)。

   d. 針對 Source (來源)，讓 Custom (自訂) 保持已選取狀態，然後輸入您希望能夠存取 Web 主控台的系統 IP 地址 (例如，`192.0.2.1`)。

   e. 選擇 Save (儲存)。

      您的代理程式現在已可接受傳入連線。

### 新增 Java 相依性

將 `activemq-client.jar` 和 `activemq-pool.jar` 套件新增到您的 Java 類別路徑。以下範例顯示 Maven 專案 `pom.xml` 檔案中的此等依存關係。

```
<dependencies>
    <dependency>
        <groupId>org.apache.activemq</groupId>
        <artifactId>activemq-client</artifactId>
        <version>5.15.8</version>
    </dependency>
    <dependency>
        <groupId>org.apache.activemq</groupId>
        <artifactId>activemq-pool</artifactId>
        <version>5.15.8</version>
    </dependency>
```

```
</dependencies>
```

如需 `activemq-client.jar` 的詳細資訊，請參閱 Apache ActiveMQ 文件中的初始組態。

> **Important**
>
> 在以下的範例程式碼中，生產者和消費者會在單一執行緒中執行。對於生產系統 (或測試代理程式執行個體容錯移轉)，請確定您的生產者和消費者在不同的主機或執行緒上執行。

## 建立訊息生產者和傳送訊息

1. 使用代理程式的端點為訊息生產者建立 JMS 集區連接工廠，然後對工廠呼叫 `createConnection` 方法。

   > **Note**
   >
   > 對於作用中/待命代理程式，Amazon MQ 會提供兩個 ActiveMQ Web 主控台 URL，但一次只有一個作用中的 URL。同樣地，Amazon MQ 為每個線路通訊協定提供兩個端點，但每個配對中一次只有一個作用中的端點。`-1` 和 `-2` 尾碼表示備援組合。如需詳細資訊，請參閱 Broker Architecture。
   >
   > 對於線路通訊協定端點，您可以允許應用程式使用容錯移轉傳輸連線到任一端點。

   ```
   // 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**
>
> 訊息生產者應一律使用 PooledConnectionFactory 類別。如需更多詳細資訊，請參閱一律使用連線集區。

2. 建立工作階段、名為 MyQueue 的佇列和訊息生產者。

```
// 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. 建立訊息字串 "Hello from Amazon MQ!"，然後傳送訊息。

```
// 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. 清除生產者。

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

## 建立訊息消費者和接收訊息

1. 使用代理程式的端點為訊息生產者建立 JMS 連線工廠，然後對工廠呼叫 `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**
>
> 訊息消費者不應使用 `PooledConnectionFactory` 類別。如需更多詳細資訊，請參閱一律使用連線集區。

2. 建立工作階段、名為 `MyQueue` 佇列和訊息消費者。

```
// 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. 開始等待訊息，並在訊息送達時接收訊息。

```
// 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**
>
> 與 AWS 傳訊服務 (例如 Amazon SQS) 不同，消費者會持續連線到代理程式。

4. 關閉消費者、工作階段和連線。

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

## 整合 ActiveMQ 代理程式與 LDAP

> **Important**
>
> RabbitMQ 代理程式不支援 LDAP 整合。

您可以使用已啟用 TLS 的下列通訊協定來存取 ActiveMQ 代理程式：

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

Amazon MQ 提供了原生 ActiveMQ 身分驗證和 LDAP 身分驗證之間的選擇，以及管理使用者許可的授權。如需與 ActiveMQ 使用者名稱和密碼相關限制的相關資訊，請參閱使用者。

若要授權 ActiveMQ 使用者和群組使用佇列和主題，您必須編輯代理程式的組態。Amazon MQ 使用 ActiveMQ 的簡單身分驗證外掛程式，將讀取和寫入限制於目的地。如需詳細資訊和範例，請參閱 一律設定授權映射 和 `authorizationEntry`。

> **Note**
>
> 目前，Amazon MQ 不支援用戶端憑證身分驗證。

主題



## 整合 LDAP 與 ActiveMQ

您可以透過儲存在輕量型目錄存取通訊協定 (LDAP) 伺服器中的登入資料驗證 Amazon MQ 使用者。您也可以新增、刪除和修改 Amazon MQ 使用者，並指派主題和佇列的許可。建立、更新和刪除代理程式之類的管理作業仍需要 IAM 登入資料，且並未與 LDAP 整合。

想要使用 LDAP 伺服器簡化和集中管理其 Amazon MQ 代理程式身分驗證和授權的客戶可以使用此功能。將所有使用者登入資料保留在 LDAP 伺服器中，可藉由提供儲存和管理這些登入資料的集中位置，節省時間和精力。

Amazon MQ 提供使用 Apache ActiveMQ JAAS 外掛程式的 LDAP 支援。Amazon MQ 也支援任何 LDAP 伺服器，例如外部程式所支援的 Microsoft Active Directory 或 OpenLDAP。如需外掛程式的詳細資訊，請參閱 Active MQ 文件的安全一節。

除了使用者之外，您還可以透過 LDAP 伺服器為特定群組或使用者指定對主題和佇列的存取權。在 LDAP 伺服器中建立代表主題和佇列的項目，然後將許可指派給特定 LDAP 使用者或群組，即可執行此動作。然後，您可設定代理程式從 LDAP 伺服器擷取授權資料。

## 先決條件

將 LDAP 支援新增至新的或現有 Amazon MQ 代理程式之前，您必須先設定服務帳戶。必須有此服務帳戶才能起始與 LDAP 伺服器的連線，而且必須具有正確的許可才能進行此連線。此服務帳戶會為您的代理程式設定 LDAP 身分驗證。任何連續的用戶端連線都會透過相同的連線進行驗證。

服務帳戶是 LDAP 伺服器中有權起始連線的帳戶。這是標準 LDAP 要求，您只必須提供一次服務帳戶登入資料。設定連線後，所有未來的用戶端連線都會透過 LDAP 伺服器進行驗證。您的服務帳戶登入資料會以加密形式安全地儲存，只有 Amazon MQ 可以存取。

若要與 ActiveMQ 整合，LDAP 伺服器上需要特定的目錄資訊樹狀結構 (DIT)。如需可清楚顯示此結構的範例`ldif` 檔案，請參閱 ActiveMQ 文件的安全一節中的將以下 LDIF 檔案匯入 LDAP 伺服器。

## LDAP 入門

若要開始使用，請瀏覽至 Amazon MQ 主控台，然後在您建立新的 Amazon MQ 或編輯現有代理程式執行個體時，選擇 LDAP authentication and authorization (LDAP 身分驗證和授權)。

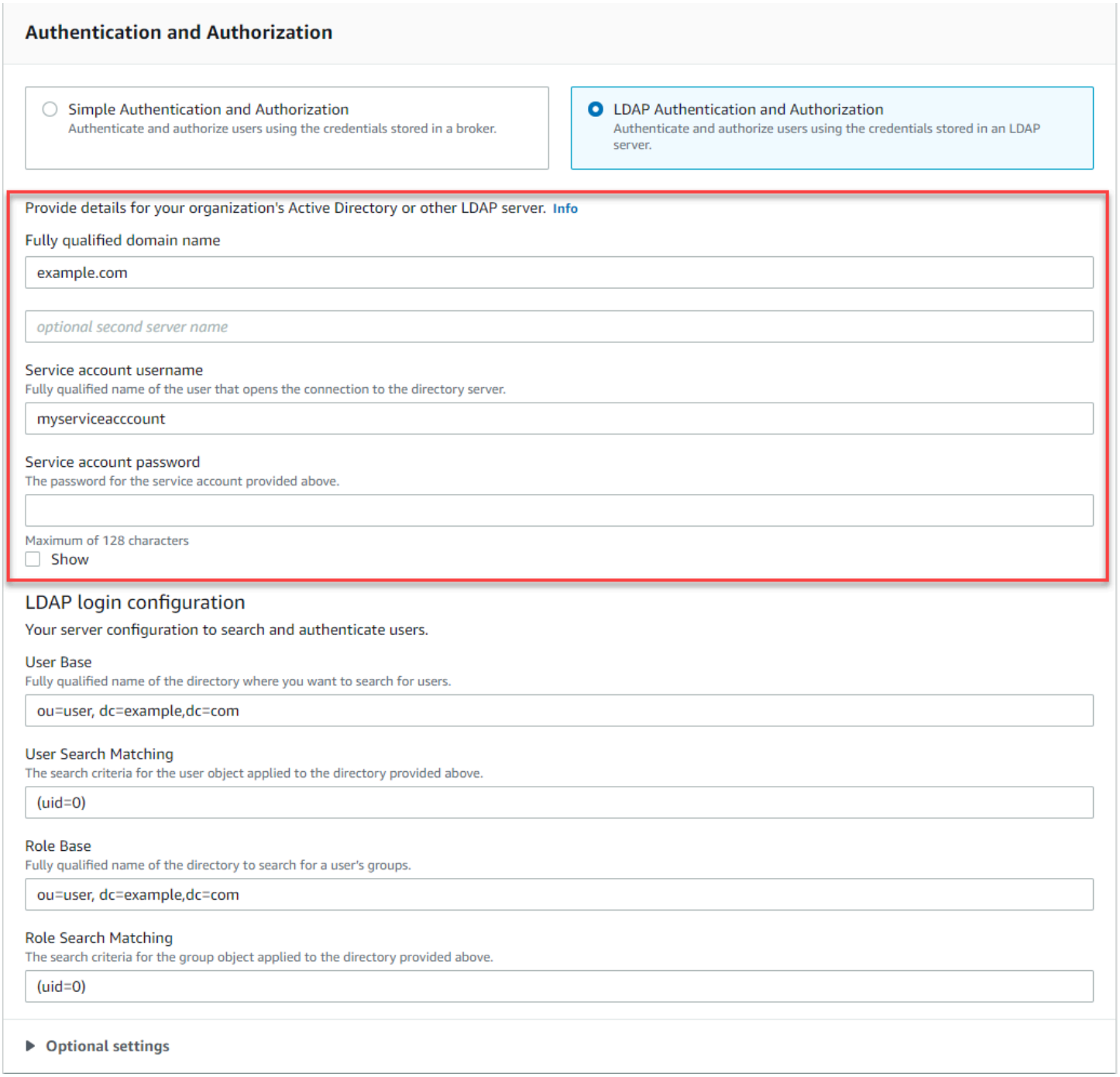
提供服務帳戶的相關資訊：

- Fully qualified domain name (完整網域名稱) 要對其發出身分驗證和授權請求的 LDAP 伺服器位置。

> **Note**
>
> 您提供的 LDAP 伺服器完整網域名稱不得包含通訊協定或連接埠號碼。Amazon MQ 會在完整網域名稱前面加上通訊協定 `ldaps`，並附加連接埠號碼 636。
> 例如，如果您提供以下完整網域：`example.com`，Amazon MQ 會使用以下 URL：`ldaps://example.com:636` 來存取您的 LDAP 伺服器。
> 為了讓代理程式主機能夠順利與 LDAP 伺服器通訊，完整網域名稱必須可公開解析。若要使 LDAP 伺服器保持私密和安全，請限制伺服器的輸入規則中的輸入流量，只允許來自代理程式 VPC 內的流量。

- Service account username (服務帳戶使用名稱) 用來執行 LDAP 伺服器初始繫連之使用者的辨別名稱。
- Service account password (服務帳戶密碼) 執行初始繫結之使用者的密碼。

下圖突顯提供這些詳細資料的位置。

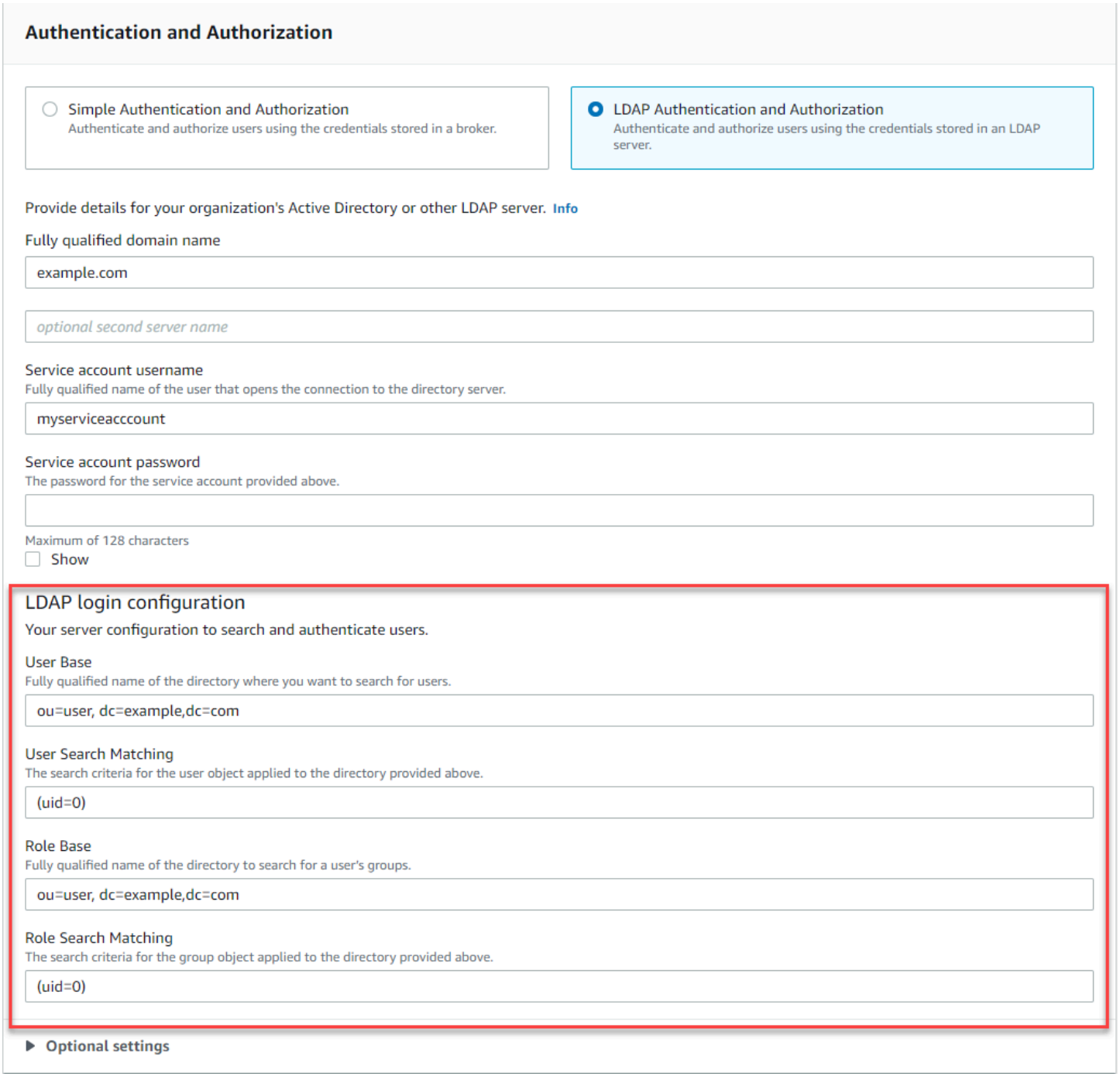



在 LDAP login configuration (LDAP 登入組態) 區段中，提供下列必要資訊：

- User Base (使用者基礎) 目錄資訊樹狀結構 (DIT) 中將搜尋使用者之節點的辨別名稱。
- User Search Matching (使用者搜尋比對) LDAP 搜尋篩選條件，將用於尋找 `userBase` 內的使用者。用戶端的使用者名稱會替換為搜尋篩選條件中的 `{0}` 預留位置。如需詳細資訊，請參閱 身分驗證 及 授權。

- Role Base (角色基礎) DIT 中將搜尋角色之節點的辨別名稱。角色可以設定為目錄中明確的 LDAP 群組項目。典型的角色項目可能包含角色名稱的一個屬性，例如一般名稱 (CN)，以及另一個屬性 (例如 `member`)，其值代表屬於角色群組之使用者的辨別名稱或使用者名稱。例如，假設組織單位 `group`，您可以提供以下辨別名稱：`ou=group,dc=example,dc=com`。
- Role Search Matching (使用者搜尋比對) LDAP 搜尋篩選條件，將用於尋找 `roleBase` 內的角色。`userSearchMatching` 比對的使用者辨別名稱會替換為搜尋篩選條件中的 `{0}` 預留位置。用戶端的使用者名稱會替換為 `{1}` 預留位置。例如，如果目錄中的角色項目包含名為 `member` 的屬性 (包含該角色中所有使用者的使用者名稱)，您可以提供以下搜尋篩選條件：`(member:=uid={1})`。

下圖突顯指定這些詳細資料的位置。




在 Optional settings (選用設定) 區段中，您可以提供下列選用資訊：

- User Role Name (使用者角色名稱) 使用者群組成員資格之使用者目錄項目中的 LDAP 屬性名稱。在某些情況下，使用者角色可能會以使用者目錄項目中的屬性值來識別。`userRoleName` 選項可讓您提供此屬性的名稱。例如，讓我們考慮以下使用者項目：

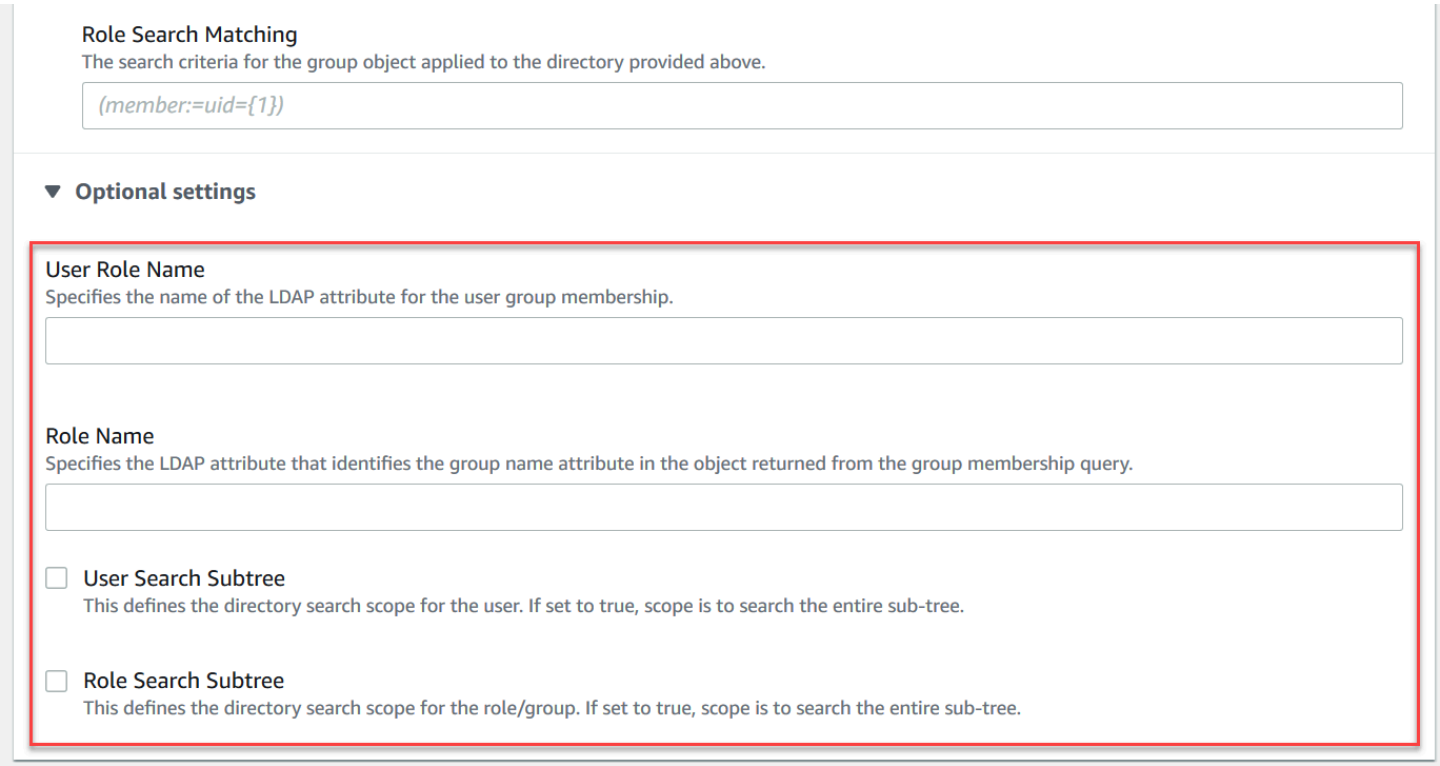
```
dn: uid=jdoe,ou=user,dc=example,dc=com
```

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

若要對上述範例提供正確的 `userRoleName`，您會指定 `memberOf` 屬性。如果身分驗證成功，則會為使用者指派角色 `role1`。

- Role Name (角色名稱) 角色項目中的群組名稱屬性，其值為該角色的名稱。例如，您可以為群組項目的一般名稱指定 `cn`。如果身分驗證成功，則會為使用者指派其所屬的每個角色項目的 `cn` 屬性值。
- User Search Subtree (使用者搜尋子樹狀結構) 定義 LDAP 使用者搜尋查詢的範圍。如果為 true，範圍會設定為搜尋 `userBase` 所定義的節點下的整個子樹樹狀結構。
- Role Search Subtree (角色搜尋子樹狀結構) 定義 LDAP 角色搜尋查詢的範圍。如果為 true，範圍會設定為搜尋 `roleBase` 所定義的節點下的整個子樹樹狀結構。

下圖突顯指定這些選用設定的位置。



## LDAP 整合的運作方式

您可考慮到兩個主要類別的集成：身分驗證結構和授權結構。

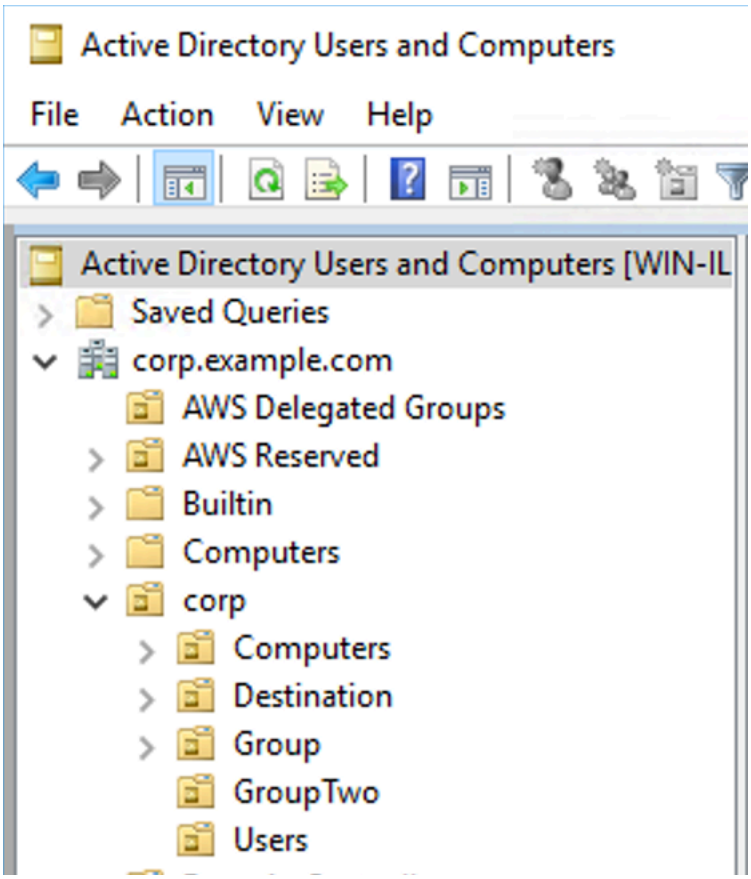
身分驗證

若要身分驗證，用戶端登入資料必須是有效的。這些登入資料會針對 LDAP 伺服器的使用者基礎中的使用者進行驗證。

提供給 ActiveMQ 代理程式的使用者基礎必須指向 DIT 中將使用者儲存在 LDAP 伺服器中的節點。例如，如果您使用 AWS Managed Microsoft AD，而且您擁有網域元件 `corp`、`example` 及 `com`，而在這些元件中您擁有組織單位 `corp` 和 `Users`，您會使用下列項目作為您的使用者基礎：

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

ActiveMQ 代理程式會在 DIT 中的這個位置搜尋使用者，以便驗證對代理程式的用戶端連線請求。



因為 ActiveMQ 原始程式碼會將使用者的屬性名稱硬式編碼為 uid，您必須確定每個使用者都已設定此屬性。為了簡單起見，您可以使用使用者的連線使用者名稱。如需詳細資訊，請參閱 activemq 原始程式碼和在 Windows Server 2016 (及後續) 版本的 Active Directory 使用者和電腦中設定 ID 映射。

若要為特定使用者啟用 ActiveMQ 主控台存取，請確定他們屬於 amazonmq-console-admins 群組。

授權

若要授權，會在代理程式組態中指定許可搜尋基礎。授權會依每個目的地 (或萬用字元、目的地集) 透過 cachedLdapAuthorizationMap 元素 (在代理程式的 activemq.xml 組態檔中找到) 進行。如需詳細資訊，請參閱快取的 LDAP 授權模組。

> **Note**
>
> 若要能夠使用您代理程式的 cachedLDAPAuthorizationMap 組態檔中的 activemq.xml 元素，您必須在透過 AWS Management Console 建立組態時，選擇 LDAP Authentication and Authorization (LDAP 身分驗證和授權) 選項，或在使用 Amazon MQ API 建立新組態時，將 authenticationStrategy 屬性設定為 LDAP。

您必須提供下列三個屬性做為 cachedLDAPAuthorizationMap 元素的一部分：

- queueSearchBase
- topicSearchBase
- tempSearchBase

> **Important**
>
> 為了防止敏感資訊直接放入代理程式的組態檔中，Amazon MQ 會阻止下列屬性使用於 cachedLdapAuthorizationMap 中：
>
> - connectionURL
> - connectionUsername
> - connectionPassword

當您建立代理程式時，Amazon MQ 會取代您透過 AWS Management Console 提供的值或 API 請求 `ldapServerMetadata` 屬性中的值，用於上述屬性。

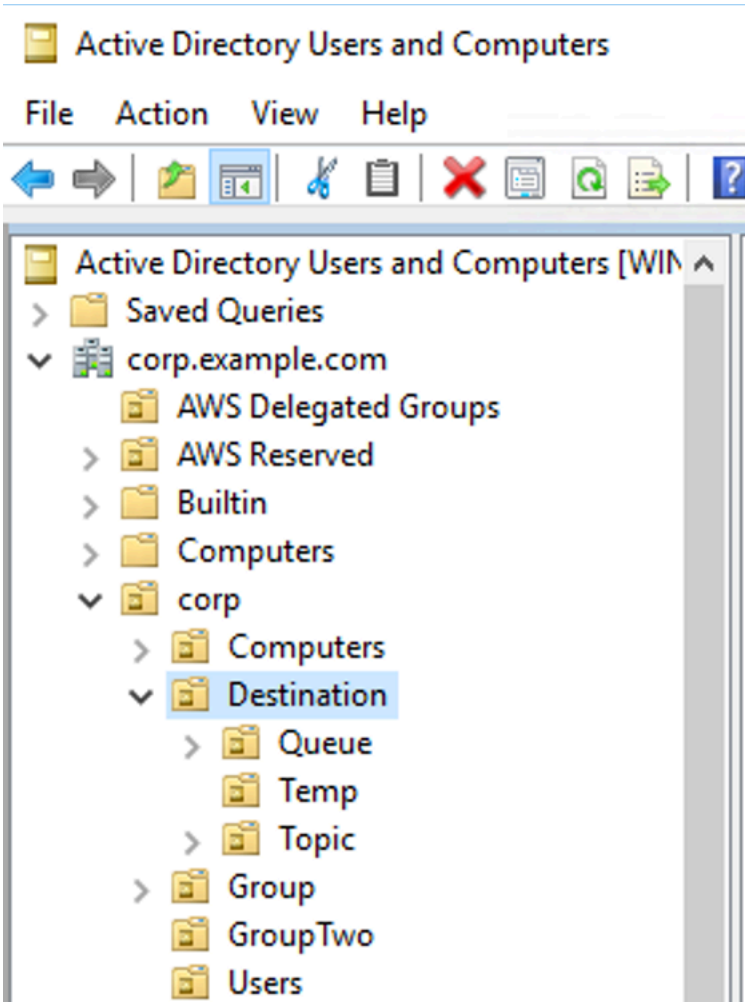
以下示範 `cachedLdapAuthorizationMap` 的正常運作範例。

```
<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>
```

這些值會識別 DIT 中指定每種目的地許可的位置。因此對上述 AWS Managed Microsoft AD 案例，使用相同的網域元件 `corp`、`example` 及 `com`，您會指定名為 `destination` 的組織單位來包含所有目的地類型。在該 OU 內，您會分別為 `queues`、`topics` 及 `temp` 目的地建立一個類型。

這表示您的佇列搜尋基礎 (針對佇列類型的目的地提供授權資訊) 會在您的 DIT 中具有以下位置：

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




同樣地，主題和暫存目的地的許可規則會位於 DIT 中的相同層級：

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

在每種目的地類型 (佇列、主題、暫存) 的 OU 中，可以提供萬用字元或特定目的地名稱。例如，若要為以前置詞 DEMO.EVENTS.$ 開頭的所有佇列提供授權規則，您可以建立以下 OU：

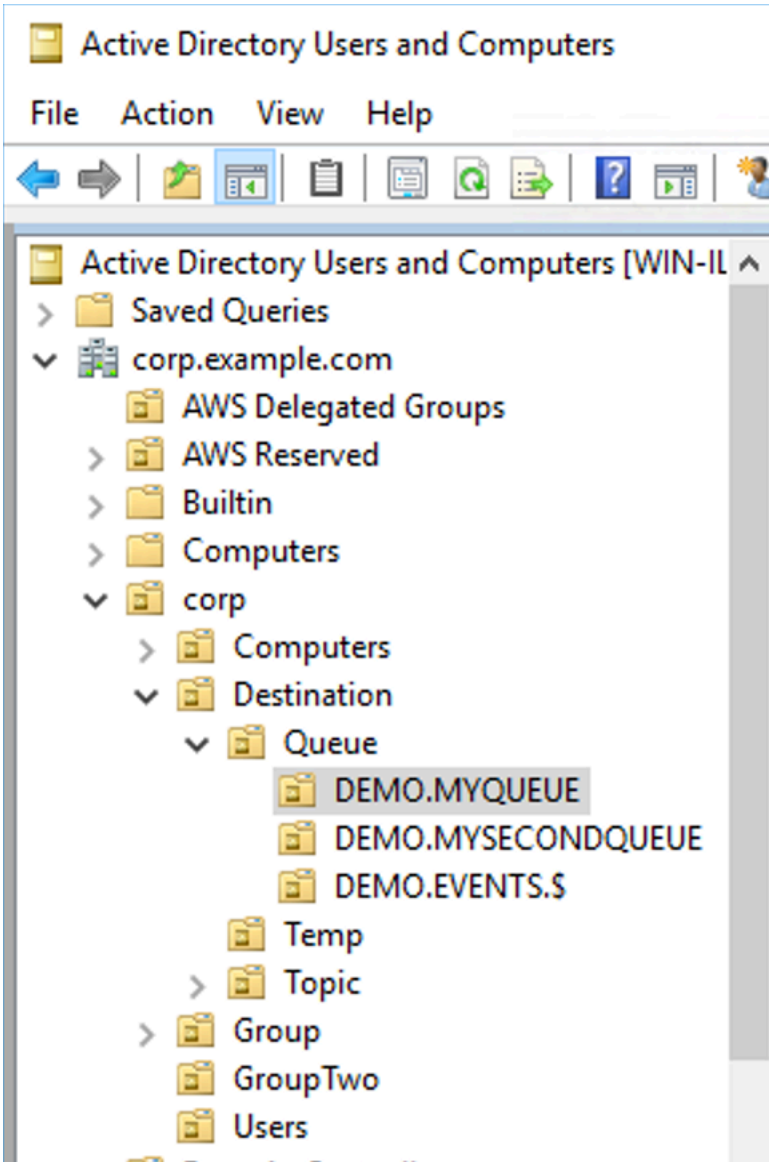
```
OU=DEMO.EVENTS.$,OU=Queue,OU=Destination,OU=corp,DC=corp,DC=example,DC=com
```

> **Note**
>
> DEMO.EVENTS.$ OU 位於 Queue OU 內。

如需 ActiveMQ 中萬用字元的詳細資訊，請參閱萬用字元

若要為特定佇列 (例如 DEMO.MYQUEUE) 提供授權規則，請指定類似下列的項目：

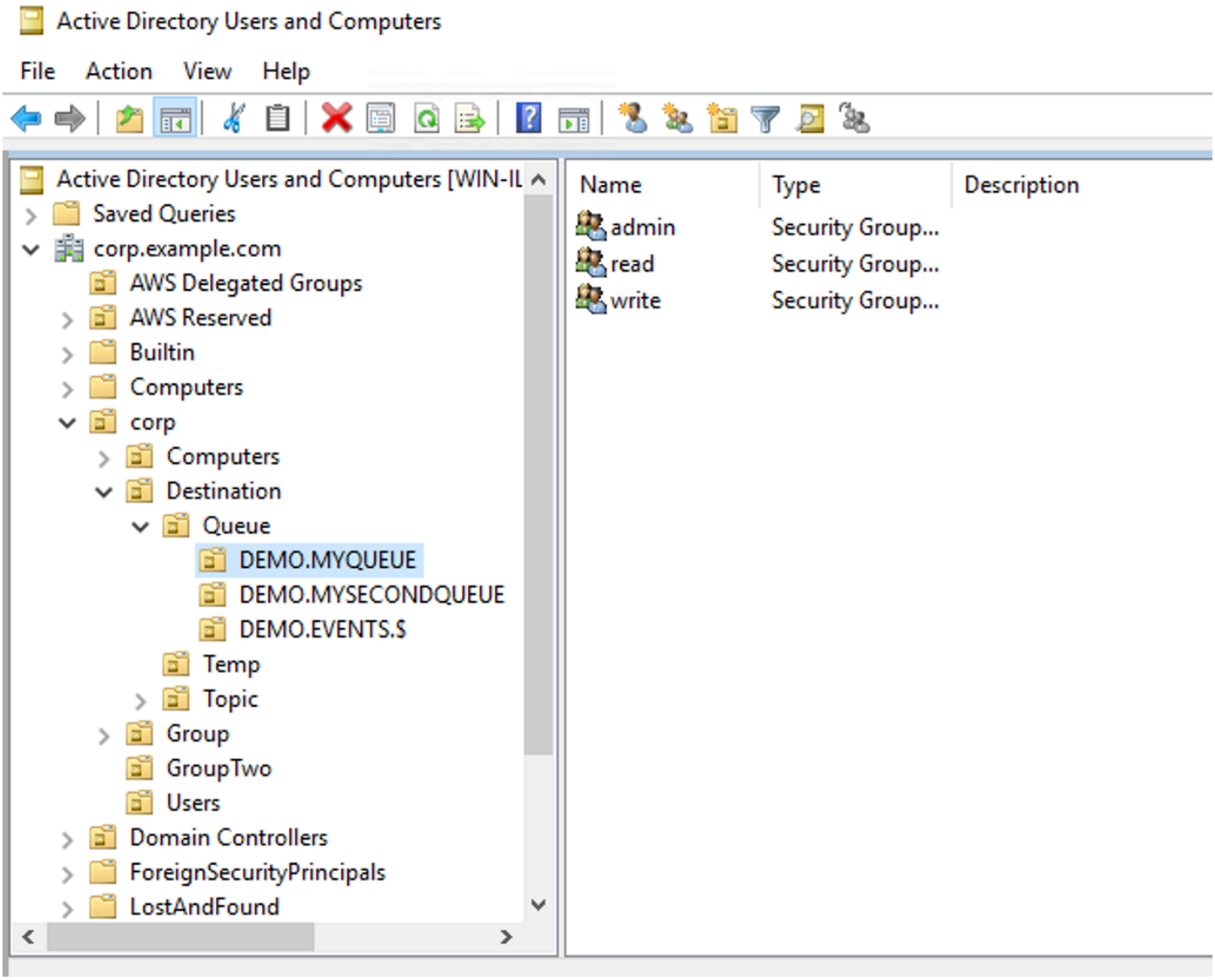
```
OU=DEMO.MYQUEUE,OU=Queue,OU=Destination,OU=corp,DC=corp,DC=example,DC=com
```



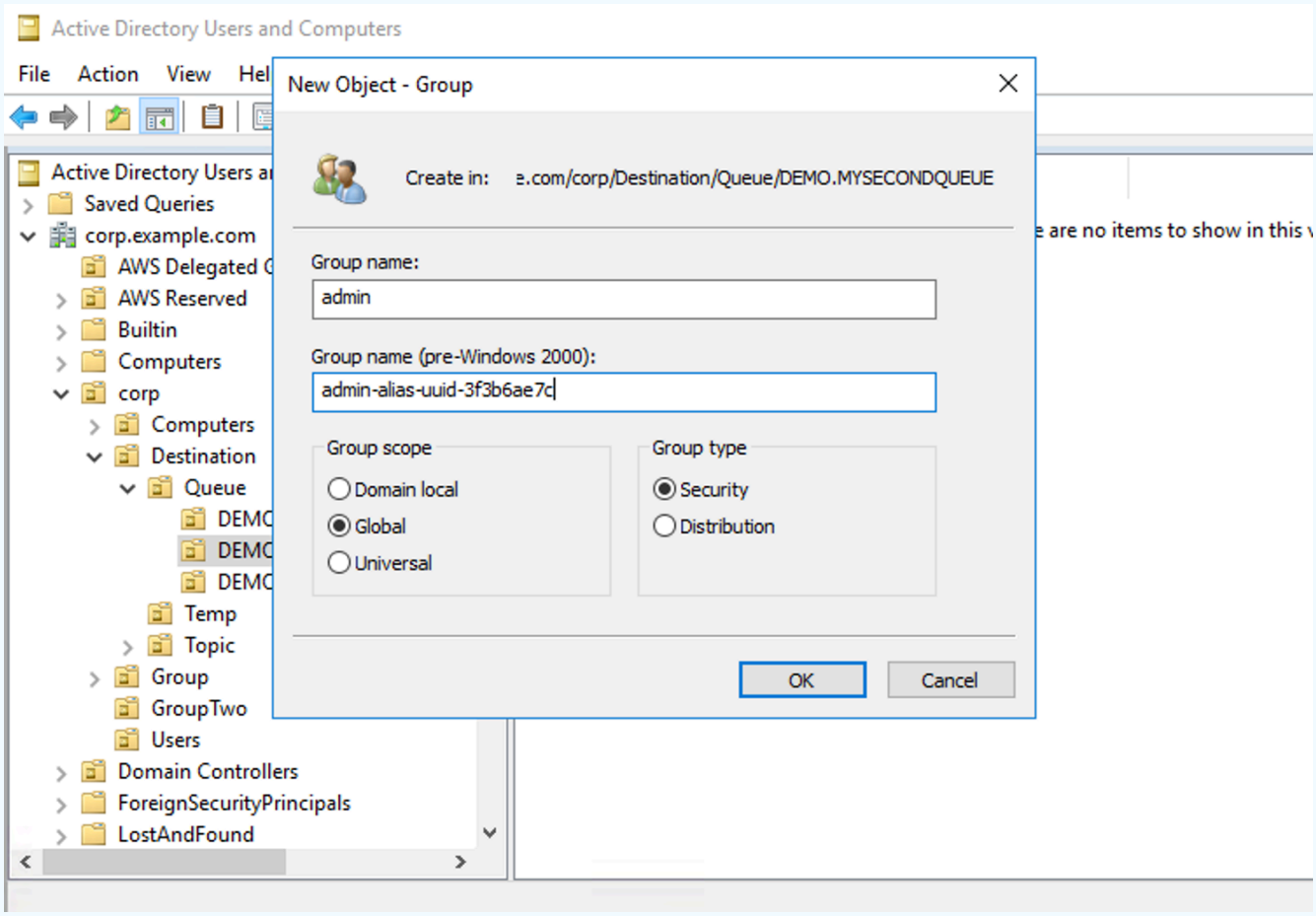

安全群組

在代表目的地或萬用字元的每個 OU 中，您必須建立三個安全群組。如同 ActiveMQ 中的所有許可，這些是讀取/寫入/管理員許可。如需各個許可允許使用者執行的動作詳細資訊，請參閱 ActiveMQ 文件中的安全。

您必須將這些安全群組命名為 `read`、`write` 及 `admin`。在這些安全群組中，您可以新增使用者或群組，而後這些使用者或群組就會有執行相關動作的許可。每個萬用字元目的地集或個別目的地都需要這些安全群組。




> **Note**
>
> 當您建立管理員群組時，群組名稱會發生衝突。發生這種衝突是因為舊版 Windows 2000 之前的規則不允許群組共用相同的名稱，即使群組位於 DIT 的不同位置亦然。Windows 2000 之前文字方塊中的值對設定沒有影響，但它必須是全域唯一的。為了避免這種衝突，您可以將 `uuid` 字尾附加至 `admin` 群組。




將使用者新增至特殊目的地的 `admin` 安全群組，可讓使用者建立和刪除該主題。將它們新增至 `read` 安全群組可讓它們從目的地讀取，而將它們新增至 `write` 群組可讓它們能夠寫入至目的地。

除了將個別使用者新增至安全群組許可之外，您也可以新增整個群組。不過，因為 ActiveMQ 再次硬式編碼群組的屬性名稱，所以您必須確定要新增的群組具有物件類別 `groupOfNames`，如 activemq 原始程式碼所示。

如果要執行這項操作，請讓使用者遵循 `uid` 的相同程序。請參閱在 Windows Server 2016 (及後續) 版本的 Active Directory 使用者和電腦中設定 ID 映射。

## 建立和管理 ActiveMQ 代理程式使用者

ActiveMQ 使用者是可以存取 ActiveMQ 代理程式的佇列和主題的人員或應用程式。您可以將使用者設定為具有特定許可。例如，您可以允許某些使用者存取 ActiveMQ Web 主控台。

群組是一個語義標籤。您可以將群組指派給使用者，並設定可供群組傳送、接收和管理特定佇列和主題的許可。

> **Note**
>
> 您無法獨立於使用者來設定群組。當您至少新增一位使用者至群組標籤時，就會建立群組標籤，並在您移除其中的所有使用者時刪除該群組標籤。

以下範例顯示如何使用 AWS Management Console 來建立、編輯和刪除 Amazon MQ 代理程式使用者。

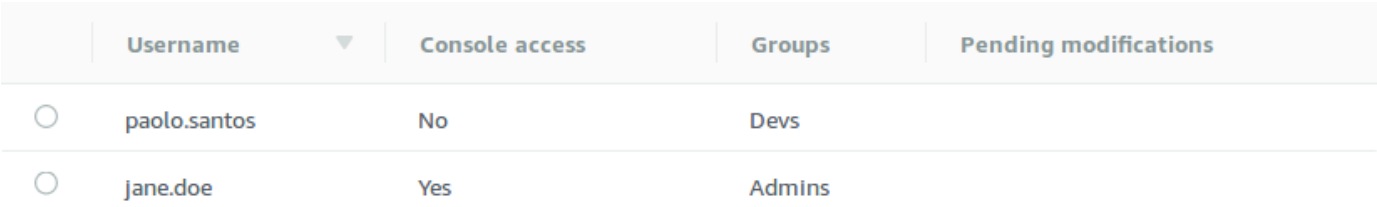
主題



## 要建立新的使用者

1. 登入 Amazon MQ 主控台。
2. 從代理程式清單，選擇您的代理程式名稱 (例如 MyBroker)，然後選擇 View details (檢視詳細資訊)。

   在 ***MyBroker*** 頁面的 Users (使用者) 區段中，會列出此代理程式的所有使用者。

3. 選擇 Create user (建立使用者)。
4. 在 Create user (建立使用者) 對話方塊中，輸入 Username (使用者名稱) 和 Password (密碼)。
5. (選用) 輸入使用者所屬的群組名稱，以逗號分隔 (例如：`Devs, Admins`)。
6. (選用) 若要讓使用者可存取 ActiveMQ Web 主控台，請選擇 ActiveMQ Web Console (ActiveMQ Web 主控台)。
7. 選擇 Create user (建立使用者)。

> **Important**
>
> 對使用者進行變更，不會立即將變更套用至使用者。若要套用變更，您必須等待下一個維護時段或重新啟動代理程式。如需更多詳細資訊，請參閱 Amazon MQ 代理程式組態生命週期。

## 編輯現有的使用者

1. 登入 Amazon MQ 主控台。
2. 從代理程式清單，選擇您的代理程式名稱 (例如 MyBroker)，然後選擇 View details (檢視詳細資訊)。

   在 ***MyBroker*** 頁面的 Users (使用者) 區段中，會列出此代理程式的所有使用者。

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

3. 選取您的登入憑證，然後選擇 編輯。

   即會顯示 Edit user (編輯使用者) 對話方塊。
4. (選用) 輸入新的 Password (密碼)。
5. (選用) 新增或移除使用者所屬的群組名稱，以逗號分隔 (例如：`Managers, Admins`)。
6. (選用) 若要讓使用者可存取 ActiveMQ Web 主控台，請選擇 ActiveMQ Web Console (ActiveMQ Web 主控台)。
7. 若要儲存使用者的變更，請選擇 Done (完成)。

> **Important**
>
> 對使用者進行變更，不會立即將變更套用至使用者。若要套用變更，您必須等待下一個維護時段或重新啟動代理程式。如需更多詳細資訊，請參閱 Amazon MQ 代理程式組態生命週期。

### 刪除現有的使用者

1. 登入 Amazon MQ 主控台。
2. 從代理程式清單，選擇您的代理程式名稱 (例如 MyBroker)，然後選擇 View details (檢視詳細資訊)。

   在 ***MyBroker*** 頁面的 Users (使用者) 區段中，會列出此代理程式的所有使用者。

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

3. 選取您的登入憑證 (例如 ***MyUser***)，然後選擇 刪除。
4. 若要確認在 Delete ***MyUser***? (刪除 MyUser？) 對話方塊中刪除使用者，請選擇 Delete (刪除)。

> **Important**
>
> 對使用者進行變更，不會立即將變更套用至使用者。若要套用變更，您必須等待下一個維護時段或重新啟動代理程式。如需詳細資訊，請參閱 Amazon MQ 代理程式組態生命週期。

# Amazon MQ for ActiveMQ 最佳實務

使用本節作為參考，快速找到在 Amazon MQ 上使用 ActiveMQ 代理程式時用於提升效能並降低輸送量成本的建議。

主題

- 連結至 Amazon MQ
- 確保有效的 Amazon MQ 效能
- 透過復原備妥的 XA 交易避免緩慢重新啟動

## 連結至 Amazon MQ

以下設計模式可提升應用程式連線至 Amazon MQ 代理程式的效益。

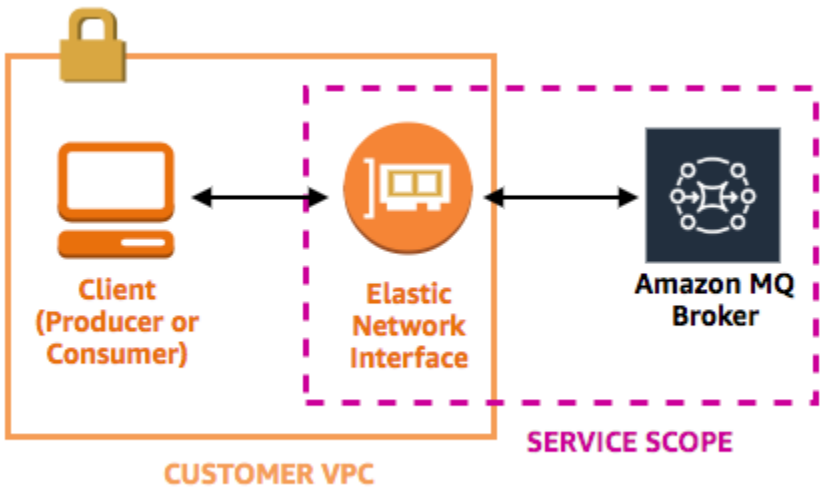
主題

- 永不修改或刪除 Amazon MQ 彈性網路界面
- 一律使用連線集區
- 一律使用容錯移轉傳輸來連接到多個代理程式端點
- 避免使用訊息選取器
- 比起耐久訂閱，更喜歡虛擬目的地
- 如果使用 Amazon VPC 對等互連，請避免使用 CIDR 範圍 10.0.0.0/16 內的用戶端 IP

## 永不修改或刪除 Amazon MQ 彈性網路界面

當您第一次建立 Amazon MQ 代理程式時，Amazon MQ 會在 Virtual Private Cloud (VPC)中您的帳戶之下佈建彈性網路界面，因此，需要一些 EC2 許可。此網路界面可讓您的用戶端 (生產者或消費者) 與 Amazon MQ 代理程式通訊。儘管此網路界面是您帳戶 VPC 的一部分，因此被視為在 Amazon MQ 的服務範圍內。

> **Warning**
>
> 您不得修改或刪除這個網路界面。修改或刪除網路界面可能導致永久遺失 VPC 與代理程式之間的連線。



## 一律使用連線集區

在具有單一生產者和單一消費者的案例 (例如 Getting Started with Amazon MQ 教學課程) 中，您可以針對每個生產者和消費者使用單一 `ActiveMQConnectionFactory` 類別。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();
```

不過，在具有多個生產者和消費者的更真實案例中，為多個生產者建立大量連線可能成本昂貴且效率不彰。在這些案例中，您應該使用 `PooledConnectionFactory` 類別，將多個生產者請求分組。例如：

> **Note**
>
> 訊息消費者不應使用 PooledConnectionFactory 類別。

```
// 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();
```

### 一律使用容錯移轉傳輸來連接到多個代理程式端點

如果您需要將應用程式連接到多個代理程式端點 (例如，當您使用作用中/待命部署模式時，或當您從內部部署訊息代理程式遷移至 Amazon MQ 時)，請使用容錯移轉傳輸，允許消費者隨機連接到其中一個。例如：

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

### 避免使用訊息選取器

您可以使用 JMS 選取器，將篩選條件附加到主題訂閱 (根據訊息的內容將訊息路由到消費者)。不過，使用 JMS 選取器會填滿 Amazon MQ 代理程式的篩選緩衝區，因而阻止它篩選訊息。

一般來說，請避免讓消費者路由訊息，因為為了讓消費者和生產者的去耦最佳化，消費者和生產者應該是暫時性。

### 比起耐久訂閱，更喜歡虛擬目的地

例如，耐久訂閱可協助確保消費者在遺失的連線還原之後，會收到所有發佈到主題的訊息。不過，使用耐久訂閱也會排除競爭消費者的使用，而且可能發生大規模的效能問題。請考慮改用虛擬目的地。

### 如果使用 Amazon VPC 對等互連，請避免使用 CIDR 範圍 **10.0.0.0/16** 內的用戶端 IP

如果您要在內部部署基礎設施與 Amazon MQ 代理程式之間設定 Amazon VPC 對等互連，則不得透過 CIDR 範圍 `10.0.0.0/16` 內的 IP 設定用戶端連線。

## 確保有效的 Amazon MQ 效能

以下設計模式可以提升 Amazon MQ 代理程式的效益和效能。

主題

- 對於具有緩慢消費者的佇列，停用並行存放和分派
- 為最佳傳輸量選擇正確的代理程式執行個體類型
- 為最佳輸送量選擇正確的代理程式儲存類型
- 正確地設定您的代理程式網路

## 對於具有緩慢消費者的佇列，停用並行存放和分派

根據預設，Amazon MQ 會針對具有快速消費者的佇列而最佳化：

- 如果消費者能夠跟得上生產者產生訊息的速率，則會將其視為快速消費者。
- 如果佇列建置未認可訊息的後端記錄，這可能造成生產者傳輸量降低，則消費者會被視為緩慢。

若要指示 Amazon MQ 針對具有緩慢消費者的佇列來最佳化，請將 `concurrentStoreAndDispatchQueues` 屬性設定為 `false`。如需範例組態，請參閱 `concurrentStoreAndDispatchQueues`。

## 為最佳傳輸量選擇正確的代理程式執行個體類型

代理程式執行個體類型的訊息傳輸量取決於應用程式的使用案例和以下因素：

- 在持久性模式中使用 ActiveMQ
- 訊息大小
- 生產者和消費者的數目
- 目的地的數目

### 了解訊息大小、延遲和輸送量之間的關係

根據您的使用案例，較大的代理程式執行個體類型可能不一定提高系統傳輸量。當 ActiveMQ 將訊息寫入到耐久性儲存時，訊息的大小決定系統的限制因素：

- 如果您的訊息小於 100 KB，則持久性儲存延遲是限制因素。
- 如果您的訊息大於 100 KB，則持久性儲存傳輸量是限制因素。

當您在持久性模式下使用 ActiveMQ 時，若有少數消費者或消費者是緩慢的，則通常會寫入至儲存。在非持久性模式下，如果代理程式執行個體的堆積記憶體已滿，則也會使用緩慢消費者寫入至儲存。

若要判斷您應用程式的最佳代理程式執行個體類型，我們建議測試不同的代理程式執行個體類型。如需詳細資訊，請參閱 Broker instance types，亦可參閱使用 JMS 基準量測 Amazon MQ 的輸送量。

### 大型代理程式執行個體類型的使用案例

當大型代理程式執行個體類型提高輸送量時，有三種常見的使用案例：

- 非持久性模式 – 當您的應用程式對在代理程式執行個體容錯移轉期間失去訊息不太敏感時 (例如，當播放運動比分時)，您通常可以使用 ActiveMQ 的非持久性模式。在此模式下，僅在代理程式執行個體的堆積記憶體已滿時，ActiveMQ 才會將訊息寫入至持久性儲存。使用非持久性模式的系統可受益於大型代理程式執行個體類型上提供的更高記憶體數量、更快 CPU，以及更快網路。
- 快速消費者 – 當作用中消費者可用，且 `concurrentStoreAndDispatchQueues` 旗標已啟用時，ActiveMQ 可讓訊息直接從生產者移到消費者，無需將訊息傳送到儲存 (甚至在持久性模式下)。如果您的應用程式可以快速耗用訊息 (或如果您可以設計讓消費者這樣做)，則您的應用程式可受益於大型代理程式執行個體類型。若要讓您的應用程式更快耗用訊息，請將消費者執行緒新增到您的應用程式執行個體，或垂直或水平擴增您的應用程式執行個體。
- 批次交易 – 當您使用持久性模式，並且為每個交易傳送多則訊息時，您可以使用大型代理程式執行個體類型來達到更高的整體訊息輸送量。如需詳細資訊，請參閱 Apache ActiveMQ 文件中的我是否應使用交易。

## 為最佳輸送量選擇正確的代理程式儲存類型

若要利用跨多個可用區域的高耐久性和複寫功能，請使用 Amazon EFS。若要利用低延遲和高輸送量，請使用 Amazon EBS。如需更多詳細資訊，請參閱 Storage。

## 正確地設定您的代理程式網路

當您建立代理程式網路時，請為您的應用程式正確地設定此項目：

- 啟用持久性模式 – 由於 (相對於同儕) 每個代理程式執行個體就像生產者或使用者，因此代理程式的網路並不提供分散式複寫的訊息。第一個做為使用者的代理程式，會接收訊息，並且將訊息儲存以持久保留。此代理程式會傳送認可訊息給生產者，並將該訊息轉傳給下一個代理程式。當第二個代理程式認可訊息的持久性時，第一個代理程式會刪除該訊息。

  如果已停用持久性模式，則第一個代理程式會發出認可訊息給生產者，而不會將訊息持久儲存。如需詳細資訊，請參閱 Apache ActiveMQ 文件中的 Replicated Message Store (複製訊息存放) 和 What is the difference between persistent and non-persistent delivery? (持久性與非持久性傳送的不同之處)。
- 不停用代理程式執行個體的建議訊息 – 如需詳細資訊，請參閱 Apache ActiveMQ 文件中的 Advisory Message (建議訊息)。
- 不使用多點傳送代理程式探索 – Amazon MQ 不支援使用多點傳送來探索代理程式。如需詳細資訊，請參閱 Apache ActiveMQ 文件中的 What is the difference between discovery, multicast, and zeroconf? (探索、多點傳送和 zeroconf 的不同之處)。

# 透過復原備妥的 XA 交易避免緩慢重新啟動

ActiveMQ 支援分散式 (XA) 交易。了解 ActiveMQ 如何處理 XA 交易，有助於避免在 Amazon MQ 進行代理程式重新啟動和容錯移轉時出現緩慢的復原時間

每次重新啟動，都會重新播放未解決的備妥 XA 交易。如果這些交易仍未解決，則其數量會隨著時間增長，大幅增加啟動代理程式所需的時間。這會影響重新啟動和容錯移轉的時間。您必須使用 `commit()` 或 `rollback()` 解決這些交易，讓效能不會隨著時間降低。

若要監控未解決的備妥 XA 交易，您可以使用 Amazon CloudWatch Logs 中的 `JournalFilesForFastRecovery` 指標。如果此數量持續增加，或是一直高於 1，您應該使用類似以下範例的程式碼來復原未解決的交易。如需更多詳細資訊，請參閱 Amazon MQ 的配額。

以下範例程式碼逐步解說備妥 XA 交易，並使用 `rollback()` 關閉它們。

```
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) {
        }
    }
}
```

在真實世界案例中，您可以針對 XA 交易管理員檢查您的備妥 XA 交易。然後，您可以決定是否要使用 `rollback()` 或 `commit()` 來處理每個備妥交易。

# Amazon MQ for ActiveMQ 跨區域資料複寫

Amazon MQ for ActiveMQ 提供跨區域資料複寫 (CRDR) 功能，該功能允許以非同步方式，將訊息從主要 AWS 區域中的主要代理程式複寫至複本區域中的複本代理程式。透過向 Amazon MQ API 發出容錯移轉請求，就可將目前的複本代理程式提升為主要代理程式角色，並將目前的主要代理程式降為複本角色。

本節提供的教學課程將說明如何在 Amazon MQ for ActiveMQ 設定跨區域資料複寫。

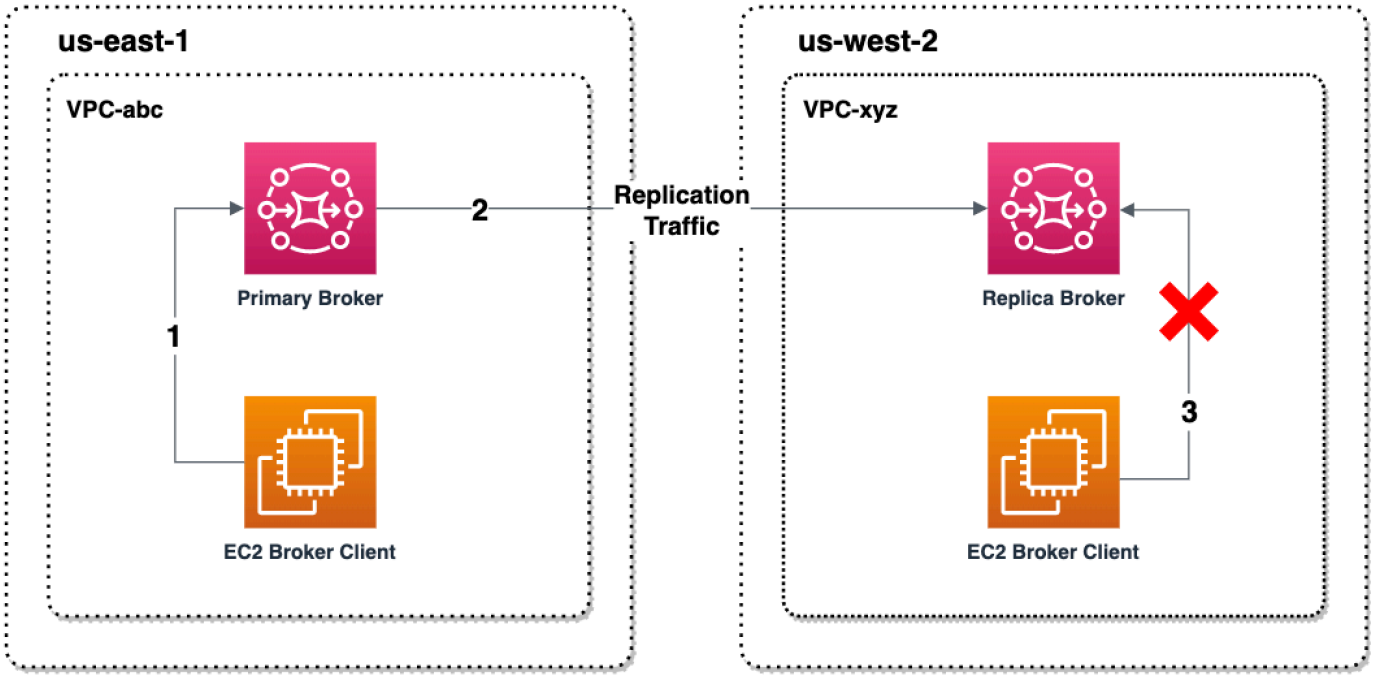
主題

- Amazon MQ 中的主要和複本代理程式
- 建立和刪除跨區域資料複寫代理程式
- 啟動切換或容錯移轉，以將複本代理程式提升為主要代理程式角色
- Amazon CloudWatch 中的跨區域資料複寫指標

## Amazon MQ 中的主要和複本代理程式

您可以建立主要和複本代理程式，以便透過非同步方式從主要 AWS 區域中的主要代理程式將資料複寫到複本區域中的複本代理程式。主要區域包含一組備援的作用中/待命代理程式配對，稱為主要代理程式。次要區域包含一組備援的作用中/待命代理程式配對，稱為複本代理程式。

下圖說明次要區域中的複本代理程式從主要區域中的主要代理程式接收非同步複寫資料。




主要和複本代理程式可作為跨區域資料復原解決方案。如果主要區域中的主要代理程式失敗，您可以藉由啟動切換或容錯移轉，將次要區域中的複本代理程式提升為主要代理程式。原本的主要代理程式會變成複本代理程式，而原本的複本代理程式則會提升為主要代理程式。如需建立主要和複本代理程式的指示，請參閱 建立和刪除跨區域資料複寫代理程式。

> **Note**
>
> 僅適用於作用中/待命代理程式。

## 建立和刪除跨區域資料複寫代理程式

使用跨區域資料複寫 (CRDR) 就可視需要在兩個 AWS 區域中的 Amazon MQ for ActiveMQ 訊息代理程式之間切換。您可以將現有代理程式指定為主要代理程式，並為此代理程式建立複本，也可以一併建立新的主要和複本代理程式。接著您可以使用 Amazon MQ `Promote` API 操作將複本代理程式提升為主要代理程式角色。如需有關主要和複本代理程式的詳細資訊，請參閱 Amazon MQ 中的主要和複本代理程式。

下列指示說明如何使用 Amazon MQ Management Console 建立和設定複本代理程式。

**主題**

## 必要條件

若要使用跨區域資料複製功能，您必須檢閱並遵守下列先決條件：

- 版本：跨區域資料複寫功能僅適用於第 5.17.6 版和更新版本的 Amazon MQ for ActiveMQ 代理程式。
- 區域：下列區域支援跨區域複寫：美國東部 (俄亥俄)、美國東部 (維吉尼亞北部)、美國西部 (奧勒岡) 及美國西部 (加利佛尼亞北部)。
- 執行個體類型：跨區域資料複寫僅適用於大小 `mq.m5.large` 以上的代理程式執行個體。
- 部署類型：跨區域資料複寫僅適用於具有多可用區域部署的運作中/待命代理程式。
- 代理程式狀態：您只能為具有 `Running` 代理程式狀態的主要代理程式建立複寫代理程式。

## 步驟 1 (選擇性)：建立新的主要代理程式

建立新的主要代理程式

1. 登入 Amazon MQ 主控台。
2. 在 Amazon MQ 主控台的「代理程式」頁面上，選擇建立代理程式。
3. 在選取代理程式引擎頁面上，選擇 Apache ActiveMQ。
4. 在選取部署和儲存頁面的部署模式和儲存類型區段中，執行下列動作：
    - 針對部署模式，選擇作用中/待命中介裝置提供高可用性。作用中/待命中介裝置提供高可用性是由兩個不同可用區域中的兩個代理程式所組成，並設定於備援組合中。這些代理程式會與您的應用程式及 Amazon EFS 同步通訊。如需更多詳細資訊，請參閱 Broker Architecture。
5. 選擇下一步。
6. 在 Configure settings (進行設定) 頁面的 Details (詳細資訊) 區段中，執行以下動作：
    a. 輸入代理程式名稱。

> **Important**
>
> 請勿在代理程式名稱中加入個人身分識別資訊 (PII) 或其他機密或敏感資訊。包括 CloudWatch Logs 在內的其他 AWS 服務可存取代理程式名稱。代理程式名稱不適用於私有或敏感資料。

b. 選擇代理程式執行個體類型 (例如，mq.m5.large)。如需更多詳細資訊，請參閱 Broker instance types。

7. 在 ActiveMQ Web 主控台存取區段中，提供使用者名稱和密碼。以下限制適用於代理程式使用者名稱和密碼：

   - 您的使用者名稱只能包含英數字元、破折號、句點、底線和波狀符號 (- . _ ~)。
   - 密碼必須至少有 12 個字元、包含至少 4 個唯一字元，而且不得包含逗號、冒號或等號 (,: =)。

> **Important**
>
> 請勿在代理程式使用者名稱中加入個人身分識別資訊 (PII) 或其他機密或敏感資訊。包括 CloudWatch Logs 在內的其他 AWS 服務可存取代理程式使用者名稱。代理程式使用者名稱不適用於私有或敏感資料。

頁面頂端的綠色閃爍列可確認 Amazon MQ 正在復原區域中建立複本代理程式。您還可以查看代理程式的 CRDR 角色和 RPO 狀態。若要關閉「CRDR 角色」和「RPO 狀態」欄，請選擇中介裝置表格右上角的齒輪圖示。然後在偏好設定頁面上，關閉「CRDR 角色」或「RPO 狀態」。

### 步驟 2：建立現有代理程式的複本

1. 在 Amazon MQ 主控台的「代理程式」頁面上，選擇建立複本中介裝置。
2. 在選擇主要中介裝置頁面上，選取要作為 CRDR 主要代理程式使用的現有代理程式。然後選擇下一步。
3. 在設定複本代理程式頁面上，使用下拉式功能表選擇複本區域。
4. 在覆寫中介裝置的 ActiveMQ 主控台使用者區段中，提供複本代理程式主控台使用者的使用者名稱和密碼。以下限制適用於代理程式使用者名稱和密碼：

   - 您的使用者名稱只能包含英數字元、破折號、句點、底線和波狀符號 (- . _ ~)。
   - 密碼必須至少有 12 個字元、包含至少 4 個唯一字元，而且不得包含逗號、冒號或等號 (,: =)。

> **Important**
>
> 請勿在代理程式使用者名稱中加入個人身分識別資訊 (PII) 或其他機密或敏感資訊。包括 CloudWatch Logs 在內的其他 AWS 服務可存取代理程式使用者名稱。代理程式使用者名稱不適用於私有或敏感資料。

5. 在代理程式之間進行橋接存取的資料複寫使用者區段中，提供將存取主要和複本代理程式之使用者的使用者名稱和密碼。以下限制適用於代理程式使用者名稱和密碼：

   - 您的使用者名稱只能包含英數字元、破折號、句點、底線和波狀符號 (- . _ ~)。
   - 密碼必須至少有 12 個字元、包含至少 4 個唯一字元，而且不得包含逗號、冒號或等號 (,: =)。

   > **Important**
   >
   > 請勿在代理程式使用者名稱中加入個人身分識別資訊 (PII) 或其他機密或敏感資訊。包括 CloudWatch Logs 在內的其他 AWS 服務可存取代理程式使用者名稱。代理程式使用者名稱不適用於私有或敏感資料。

   進行任何其他設定。然後選擇下一步。
6. 在檢閱並建立頁面上，檢閱複本代理程式詳細資訊。然後選擇建立複本中介裝置。
7. 接著重新啟動主要代理程式。這樣做也會重新啟動複本代理程式。如需有關重新啟動代理程式的指示，請參閱 Rebooting a Broker。

如需為 ActiveMQ 代理程式進行其他設定的詳細資訊，請參閱 建立並連線至 ActiveMQ 代理程式

## 刪除 CRDR 代理程式

若要刪除主要或複本 CRDR 代理程式，您必須先取消配對，然後重新啟動代理程式。下列指示說明如何使用 AWS Management Console 取消配對及重新啟動代理程式。

1. 在中介裝置頁面上，選取要取消配對的 CRDR 代理程式，然後選擇編輯。
2. 在代理程式編輯頁面的資料覆寫區段中，選擇取消代理程式配對。
3. 在快顯視窗中輸入「取消配對」，以確認您的選擇。然後選擇取消代理程式配對。

4. 接著重新啟動已取消配對的主要代理程式。這樣做也會重新啟動複本代理程式。如需有關重新啟動代理程式的指示，請參閱 Rebooting a Broker。主要代理程式重新啟動後，兩個代理程式都已取消配對，並且可分別刪除。若要刪除您的代理程式，請參閱 Deleting a broker。

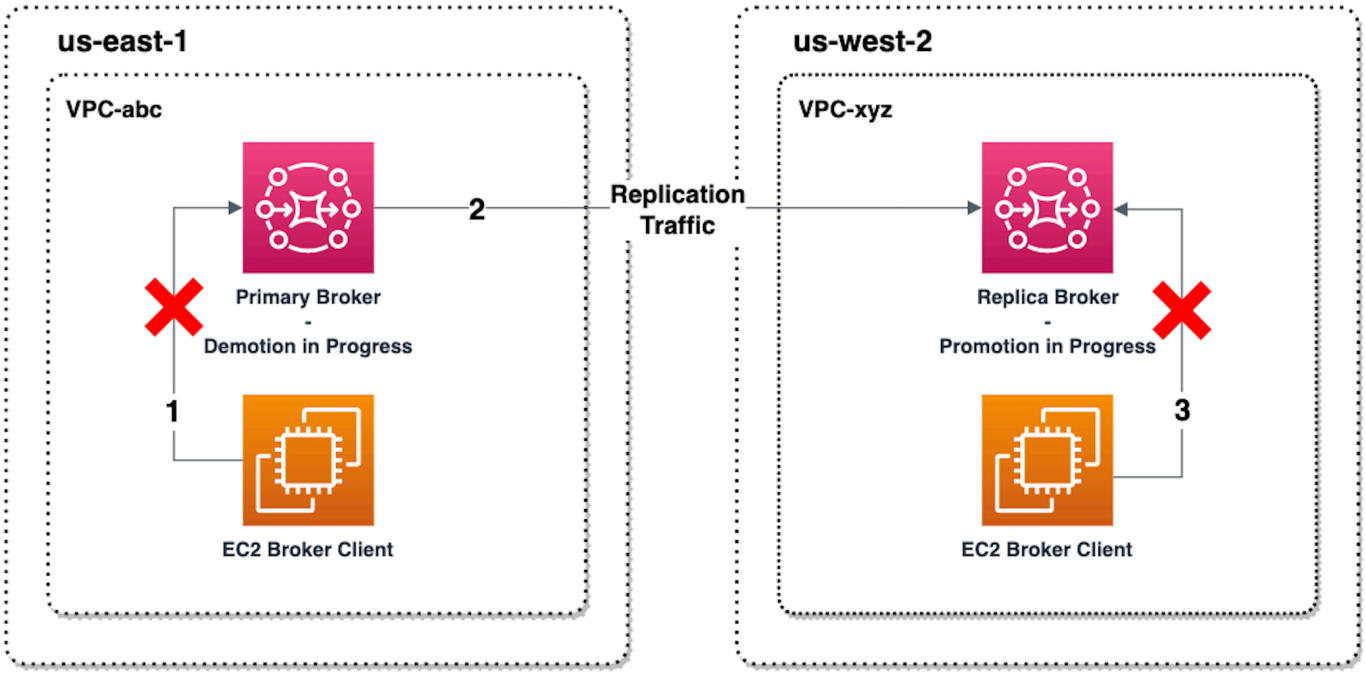
## 啟動切換或容錯移轉，以將複本代理程式提升為主要代理程式角色

當您想要將複本代理程式提升為主要代理程式角色時，可以啟動切換或容錯移轉。當您提升複本代理程式時，主要代理程式會降為複本代理程式角色。

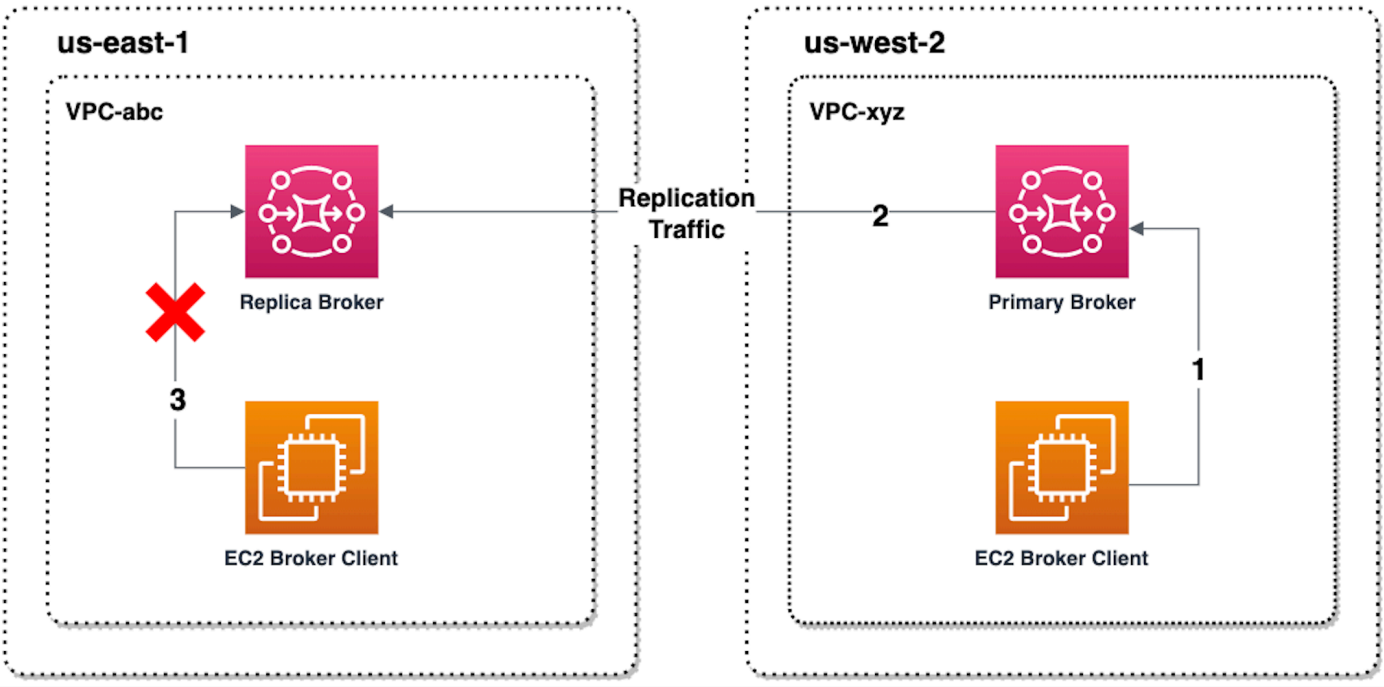
切換會將一致性視為優先於可用性。當此容錯移轉操作完成時，就可保證代理程式擁有一致的狀態。使用切換時，在建立代理程式間一致性的情況下，可能會有一段時間用戶端連線都無法使用代理程式。複本提升時，兩個代理程式會立即擁有相同的狀態。切換是否成功，取決於兩個地區的運作狀態和區域間網路是否成功。

容錯移轉會將可用性視為優先於一致性。當此容錯移轉操作完成時，並不能保證代理程式擁有一致的狀態。使用容錯移轉時，可保證複本代理程式會立即提供用戶端流量服務，而不需等待任何複寫資料同步化，也不需等待主要代理程式收到關閉訊號。容錯移轉既非取決於原始主要區域的運作狀態，也非取決於區域間網路是否成功。

下圖說明的切換，是在排空複寫佇列且代理程式狀態已同步時，代理程式均不接受用戶端連線的情況。在此程序中，主要代理程式的 VPC 中的用戶端無法在容錯移轉進行中且主要代理程式降為複本時，進一步產生狀態變更。當複寫佇列已排空且兩個代理程式達到一致的狀態時，在容錯移轉操作完成且複本代理程式提升為主要代理程式之前，複本代理程式的 VPC 中的用戶端無法連線到複本代理程式。




下圖說明切換程序完成後的代理程式狀態。原始複本代理程式現已提升為主要代理程式角色，並且會接受用戶端連線。用戶端可從代理程式產生和使用資料。



### 使用主控台提升複本代理程式

若要使用切換或容錯移轉提升複本代理程式，請在 Amazon MQ 主控台中執行下列步驟。

> **Note**
>
> 您無法在主要代理程式上啟動切換或容錯移轉。

1. 切換至複本代理程式所在的區域。從「代理程式」表格，選取您要提升為主要代理程式的現有複本代理程式。
2. 在代理程式詳細資訊頁面上，執行以下作業：
    1. 選取升級覆寫。
    2. 在快顯視窗中，選擇轉換或容錯移轉。
    3. 在文字方塊中輸入「確認」以確認您的選擇。
    4. 選擇確認。

啟動容錯移轉之後，代理程式的狀態會變更為容錯移轉進行中。容錯移轉完成時，「代理程式」頁面頂端的藍色進度列會變成綠色。

> **Note**
>
> 只有在建立複本代理程式時才會複寫組態。不會複製之後的任何更新。

## Amazon CloudWatch 中的跨區域資料複寫指標

Amazon MQ for ActiveMQ 跨區域資料複寫功能提供了多種指標，可用來維護主要和複本代理程式的可靠性、可用性和效能。在複寫程序期間，次要區域中的複本代理程式會接收來自主要區域中主要代理程式的非同步複寫資料。如果主要區域中的主要代理程式失敗，您可以藉由啟動切換或容錯移轉，將次要區域中的複本代理程式提升為主要代理程式。如需檢視 Amazon CloudWatch 中各種指標的指示，請參閱 存取 Amazon MQ 的 CloudWatch 指標。

### CRDR 時間戳記

下列時間戳記說明 Amazon CloudWatch 中各種指標的計算方式。資料複製程序中有五個時間戳記：

- 目前觀測時間 (TCO)：目前的時間點。
- 建立時間 (TC)：主要代理程式在複寫佇列上建立事件的時間點。主要和複本代理程式兩者都適用。
- 傳遞時間 (TD)：事件成功傳遞至複本代理程式的時間點。僅適用於複本代理程式。
- 處理時間 (TP)：複本代理程式成功處理事件的時間點。僅適用於複本代理程式。
- 確認時間 (TA)：主要代理程式成功確認事件的時間點。僅適用於主要代理程式。

## 利用 CRDR CloudWatch 指標來預估切換/容錯移轉效能

預設情況下，Amazon MQ 為您的代理程式啟用指標。您可以透過存取 Amazon CloudWatch 主控台，或使用 CloudWatch API，檢視您的代理程式指標。下列指標有助於了解 CRDR 代理程式的複寫和切換/容錯移轉效能：

| Amazon MQ CloudWatch 指標 | 使用 CRDR 的原因 |
| --- | --- |
| `TotalReplicationLag` | 主要代理程式上最後一個未確認事件的 TA 和 TC 之間的預估時間。 |
| `ReplicationLag` | 複本代理程式上最後一個未確認事件的 TP 和 TC 之間的預估時間。 |
| `PrimaryWaitTime` | 主要代理程式上最後一個已處理事件的 TCO 和 TC 之間的預估時間。 |
| `ReplicaWaitTime` | 複本代理程式上最後一個已處理事件的 TCO 和 TP 之間的預估時間。 |
| `QueueSize` | 主要代理程式上複寫佇列中未確認事件的總數。 |

`TotalReplicationLag` 和 `ReplicationLag` 說明主要和複本代理程式之間延遲的複寫。這兩個指標也可用於預估進行中的切換或容錯移轉操作還有多久時間會完成。

`PrimaryWaitTime` 和 `ReplicaWaitTime` 可用於找出複寫程序中任何持續發生的問題。如果指標的值持續增加，可能表示複寫程序已降級或暫停。複寫變慢的原因可能包括網路分割、代理程序啟動，以及復原時間長等問題。

# Amazon MQ for ActiveMQ 中的配額

本主題列出 Amazon MQ 中的配額。您可以針對特定 AWS 帳戶變更下列許多配額。若要請求提高限制，請參閱 Amazon Web Services 一般參考 中的 AWS Service Quotas。即使套用上限，更新的限制也不會顯示。如需在 Amazon 中檢視目前連線限制的詳細資訊 CloudWatch，請參閱使用 Amazon 監控 Amazon MQ 代理程式。 CloudWatch

> **Note**
>
> 如需 Amazon MQ for RabbitMQ 的配額，請參閱 Amazon MQ for RabbitMQ 的配額。

主題



## 中介裝置

下表列出與 Amazon MQ for ActiveMQ 代理程式相關的配額。

| 限制 | 描述 |
| --- | --- |
| 代理程式名稱 | • 在經紀人地區和您的 AWS 帳戶中必須是唯一的。<br>• 長度必須介於 1 至 50 個字元之間。<br>• |

| 限制 | 描述 |
| --- | --- |
| | 必須僅包含 ASCII 可列印字元集中指定的字元。<br>• 只能包含英數字元、破折號、句點、底線和波狀符號 (- . _ ~)。 |
| 每區域的代理程式數目 | 50 |
| 較小型代理程式每個通訊協定的線路層級連線 | `mq.*.micro` 執行個體類型代理程式為 300 個。 |
| 較大型代理程式每個通訊協定的線路層級連線 | `mq.*.*large` 執行個體類型代理程式為 2,000 個。 |
| 網路連接器的數量 | 20 |
| 每個代理程式的安全群組數 | 5 |
| 中監視的 ActiveMQ 目的地 (佇列和主題) CloudWatch | CloudWatch 僅監控前 1000 個目的地。 |
| RabbitMQ 目的地 (佇列) 監控於 CloudWatch | CloudWatch 僅監控前 500 個目的地，按消費者數量排序。 |
| 每個代理程式的標籤 | 50 |

## 組態

下表列出與 Amazon MQ for ActiveMQ 組態相關的配額。

| 限制 | 描述 |
| --- | --- |
| 組態名稱 | • 長度必須介於 1 至 150 個字元之間。<br>• 必須僅包含 ASCII 可列印字元集中指定的字元。 |

| 限制 | 描述 |
| --- | --- |
| | • 只能包含英數字元、破折號、句點、底線和波狀符號 (- . _ ~)。 |
| 每個組態的修訂數 | 300 |

## 使用者

下表列出與 Amazon MQ for ActiveMQ 代理程式使用者相關的配額。

| 限制 | 描述 |
| --- | --- |
| 使用者名稱 | • 長度必須介於 1 至 100 個字元之間。<br>• 必須僅包含 ASCII 可列印字元集中指定的字元。<br>• 只能包含英數字元、破折號、句點、底線和波狀符號 (- . _ ~)。<br>• 不得包含逗號 (,)。 |
| 密碼 | • 長度必須介於 12 至 250 個字元之間。<br>• 必須僅包含 ASCII 可列印字元集中指定的字元。<br>• 必須包含至少 4 個唯一字元。<br>• 不得包含逗號 (,)。 |
| 每個代理程式的使用者 (簡單身分驗證) | 250 |
| 每個使用者的群組 (簡單身分驗證) | 20 |

## 資料儲存體

下表列出與 Amazon MQ for ActiveMQ 資料儲存相關的配額。

| 限制 | 描述 |
| --- | --- |
| 每個較小代理程式的儲存容量 | `mq.*.micro` 執行個體類型代理程式為 20 GB。如需 Amazon MQ 執行個體類型的詳細資訊，請參閱 Broker instance types。 |
| 每個較大代理程式的儲存容量 | `mq.*.*large` 執行個體類型代理程式為 200 GB。如需 Amazon MQ 執行個體類型的詳細資訊，請參閱 Broker instance types。 |
| Amazon EBS 支援的每個代理程式的任務排程器使用量限制 | 50 GB。如需任務排程器使用量的詳細資訊，請參閱 Apache ActiveMQ API 文件中的 `JobSchedulerUsage`。 |
| 每個較小代理程式的暫時儲存容量。 | `mq.*.micro` 執行個體類型代理程式為 5 GB。 |
| 每個較大代理程式的暫時儲存容量。 | `mq.*.*large` 執行個體類型代理程式為 50 GB。 |

## API 調節

下列節流配額會在所有 Amazon MQ API 中彙總每個 AWS 帳戶，以維護服務頻寬。如需 Amazon MQ API 的詳細資訊，請參閱 Amazon MQ REST API 參考。

> **Important**
>
> 這些配額不適用於 Amazon MQ for ActiveMQ 或 Amazon MQ for RabbitMQ 代理程式傳訊 API。例如，Amazon MQ 不會調節訊息的傳送或接收。

| API 高載限制 | API 速率限制 |
| --- | --- |
| 100 | 15 |

# 使用 Amazon MQ for RabbitMQ

Amazon MQ 可讓您利用符合您需求的運算和儲存資源輕鬆地建立訊息代理程式。您可以使用 AWS Management Console、 Amazon MQ REST API 或 AWS Command Line Interface 來建立、管理和刪除代理程式。

本節描述 ActiveMQ 和 RabbitMQ 引擎類型之訊息代理程式的基本元素、列出可用的 Amazon MQ 代理程式執行個體類型及其狀態，並提供代理程式架構和組態選項的概觀。

若要了解 Amazon MQ REST API，請參閱 Amazon MQ REST API 參考。

主題



## RabbitMQ 引擎

本節說明 RabbitMQ 代理程式的基本元素及其支援外掛程式，並提供 Amazon MQ 上 RabbitMQ 代理程式架構選項的概觀。

主題



### 基本元素

本節簡介了解 Amazon MQ 上 RabbitMQ 所需的重要概念。

主題

- 中介裝置執行個體類型
- 組態
- 使用者
- 外掛程式
- 政策

## 代理程式

代理程式是在 Amazon MQ 上執行的訊息代理程式環境。這是 Amazon MQ 的基本建置區塊。代理程式執行個體類別 (`m5`、`t3`) 和大小 (`large`、`micro`) 的合併說明是代理程式執行個體類型 (例如，`mq.m5.large`)。如需詳細資訊，請參閱 Broker instance types。

- 單一執行個體代理程式是由 Network Load Balancer (NLB) 後面的一個可用區域中的一個代理程式組成。代理程式會與您的應用程式以及 Amazon EBS 儲存磁碟區進行通訊。
- 叢集部署是 Network Load Balancer 後面的三個 RabbitMQ 代理程式節點的邏輯分組，每個節點共用使用者、佇列，以及跨多個可用區域 (AZ) 的分散式狀態。

如需詳細資訊，請參閱 代理程式架構。

當新版的 RabbitMQ 引擎發行時，您可以啟用自動次要版本升級，以升級到代理程式引擎的新次要版本。自動升級會發生於由星期幾、一天中的時間 (24 小時制) 和時區 (預設為 UTC) 所定義的維護時段期間。

### 支援的通訊協定

您可以使用 RabbitMQ 支援的任何程式設計語言，並為下列通訊協定啟用 TLS，以存取 RabbitMQ 代理程式：

- AMQP (0-9-1)

### 接聽程式連接埠

Amazon MQ 受管 RabbitMQ 代理程式支援將下列接聽程式連接埠用於透過 `amqps` 的應用程式層級連線，以及使用 RabbitMQ Web 主控台和管理 API 的用戶端連線。

- 接聽連接埠 `5671` - 用於透過安全 AMQP URL 進行的連線。例如，假設有代理程式 ID 為 `b-c8352341-ec91-4a78-ad9c-a43f23d325bb` 的代理程式，部署在 `us-west-2` 地區中，以

下是代理程式的完整 amqp URL：`b-c8352341-ec91-4a78-ad9c-a43f23d325bb.mq.us-west-2.amazonaws.com:5671`。

- 接聽程式連接埠 443 和 15671 - 這兩個接聽程式連接埠可以互換使用，以透過 RabbitMQ Web 控制台或管理 API 存取代理程式。

Attributes

RabbitMQ 代理程式具有多個屬性：

- 名稱。例如，`MyBroker`。
- ID。例如 `b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9`。
- Amazon 資源名稱 (ARN) 例如 `arn:aws:mq:us-east-2:123456789012:broker:MyBroker:b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9`。
- RabbitMQ Web 主控台 URL。例如 `https://b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-east-2.amazonaws.com`。

  如需詳細資訊，請參閱 RabbitMQ 文件中的 RabbitMQ Web 主控台。
- 安全的 AMQP 端點。例如 `amqps://b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-east-2.amazonaws.com`。

如需代理程式屬性的完整清單，請參閱 Amazon MQ REST API 參考中的以下各節：

- REST 操作 ID：中介裝置
- REST 操作 ID：中介裝置
- REST 操作 ID：中介裝置重新開機

## 代理程式預設值

當您建立 Amazon MQ for RabbitMQ 代理程式時，Amazon MQ 會套用一組預設政策和虛擬主機限制，以最佳化代理程式的效能。Amazon MQ 只會將虛擬主機限制套用至預設 (/) 虛擬主機。Amazon MQ 不會將預設政策套用至新建立的虛擬主機。我們建議為所有新的和現有代理程式保留這些預設值。但是，您可以隨時修改、覆寫或刪除這些預設值。

Amazon MQ 會根據您在建立代理程式時選擇的執行個體類型和代理程式部署模式建立政策和限制。預設政策會根據部署模式來命名，如下所示：

- 單一執行個體 – `AWS-DEFAULT-POLICY-SINGLE-INSTANCE`

- 叢集部署 – AWS-DEFAULT-POLICY-CLUSTER-MULTI-AZ

對於單一執行個體代理程式，Amazon MQ 會將政策優先順序值設定為 0。若要覆寫預設優先順序值，您可以建立具有較高優先順序值的自訂政策。對於叢集部署，Amazon MQ 會將優先順序值設定為 1 作為代理程式預設值。若要為叢集建立自己的自訂政策，請指派大於 1 的優先順序值。

> **Note**
>
> 在叢集部署中，需要有 ha-mode 和 ha-sync-mode 代理程式政策，才能達到傳統鏡像和高可用性 (HA)。
> 如果您刪除預設 AWS-DEFAULT-POLICY-CLUSTER-MULTI-AZ 政策，則 Amazon MQ 會使用優先順序值為 0 的 ha-all-AWS-OWNED-DO-NOT-DELETE 政策。這可確保所需的 ha-mode 和 ha-sync-mode 政策仍然有效。如果您建立自己的自訂政策，Amazon MQ 會自動將 ha-mode 和 ha-sync-mode 附加至您的政策定義。

主題



政策和限制說明

下列清單說明 Amazon MQ 套用至新建代理程式的預設政策和限制。max-length、max-queues 及 max-connections 的值會根據代理程式的執行個體類型和部署模式而有所不同。這些值列在 建議的預設值 區段中。

- **queue-mode: lazy** (政策) – 啟用延遲佇列。依預設，佇列會保留訊息的記憶體內快取，讓代理程式能夠盡快將訊息傳遞給消費者。這可能會導致代理程式記憶體不足，並引發高記憶體警示。延遲佇列會儘早嘗試將訊息移至磁碟。這表示在正常操作條件下，記憶體中保存的訊息較少。使用延遲佇列，Amazon MQ for RabbitMQ 可支援更大的傳訊負載和更長的佇列。請注意，在某些使用案例中，具有延遲佇列的代理程式執行速度可能會稍微慢一些。這是因為訊息會從磁碟移至代理程式，而不是從記憶體內快取傳送訊息。

  > **部署模式**
  >
  > 單一執行個體、叢集

- **`max-length: number-of-messages`** (政策) – 設定佇列中的訊息數量限制。在叢集部署中，此限制會防止在代理程式重新啟動的情況下或在維護時段之後暫停佇列同步處理。

  部署模式

  叢集

- **`overflow: reject-publish`** (策略) – 強制採用 `max-length` 政策的佇列，在佇列中的訊息數量達到 `max-length` 值之後，拒絕新訊息。若要確保佇列處於溢位狀態時訊息不會遺失，將訊息發佈至代理程式的用戶端應用程式必須實作發行者確認。如需實作發行者確認的相關資訊，請參閱 RabbitMQ 網站上的 Publisher Confirms (發行者確認)。

  部署模式

  叢集

- **`max-queues: number-of-queues-per-vhost`** (虛擬主機限制) – 設定代理程式中佇列數目的限制。類似於 `max-length` 政策定義，限制叢集部署中的佇列數目可避免在代理程式重新啟動或維護時段後暫停佇列同步處理。限制佇列也可避免將過多的 CPU 使用量用於維護佇列。

  部署模式

  單一執行個體、叢集

- **`max-connections: number-of-connections-per-vhost`** (虛擬主機限制)– 設定與代理程式的用戶端連線數目限制。根據建議的值限制連線數目，可防止過度的代理程式記憶體使用量，進而導致代理程式引發高記憶體警示和暫停操作。

  部署模式

  單一執行個體、叢集

### 建議的預設值

> **Note**
>
> `max-length` 和 `max-queue` 預設限制會根據 5 kB 的平均訊息大小進行測試和評估。如果您的訊息明顯大於 5 kB，則需要調整並降低 `max-length` 和 `max-queue` 限制。

下表列出新建代理程式的預設限制值。Amazon MQ 會根據代理程式的執行個體類型和部署模式套用這些值。

| 執行個體類型 | 部署模式 | **max-length** | **max-queues** | **max-connections** |
|---|---|---|---|---|
| t3.micro | 單一執行個體 | N/A | 500 | 500 |
| m5.large | 單一執行個體 | N/A | 20,000 | 4,000 |
| | 叢集 | 8,000,000 | 4,000 | 15,000 |
| m5.xlarge | 單一執行個體 | N/A | 30,000 | 8,000 |
| | 叢集 | 9,000,000 | 5,000 | 20,000 |
| m5.2xlarge | 單一執行個體 | N/A | 60,000 | 15,000 |
| | 叢集 | 10,000,000 | 6,000 | 40,000 |
| m5.4xlarge | 單一執行個體 | N/A | 150,000 | 30,000 |
| | 叢集 | 12,000,000 | 10,000 | 100,000 |

## 中介裝置執行個體類型

> **Important**
>
> 您不能將代理程式從 `mq.m5.` 執行個體類型降級至 `mq.t3.micro` 執行個體類型。

> **Important**
>
> 您目前無法在 euc1-az2 可用區域中建立 `t2.micro`、`m4.large` ，或 `m5.*` 代理程式。

| 執行個體類型 | vCPU | 記憶體 (GiB) | 網路效能 | 備註 |
| --- | --- | --- | --- | --- |
| `mq.t3.micro` | 2 | 1 | 低 | 使用 `mq.t3.micro` 執行個體類型進行 Amazon MQ 的基本評估。此執行個體類型 (僅限單一執行個體代理程式) 符合 AWS 免費方案的資格。<br><br>**Important**<br>`mq.t3.micro` 執行個體類型不支援叢集部署。 |
| `mq.m5.large` | 2 | 8 | 高 | 將 `mq.m5.large` 執行個體用於一般開發、測試及生產工作負載。 |
| `mq.m5.xlarge` | 4 | 16 | 高 | 將 `mq.m5.xlarge` 、`mq.m5.2xlarge` 和 `mq.m5.4xlarge` 執行個體 |
| `mq.m5.2xlarge` | 8 | 32 | 高 | |

| 執行個體類型 | vCPU | 記憶體 (GiB) | 網路效能 | 備註 |
| --- | --- | --- | --- | --- |
| mq.m5.4xlarge | 16 | 64 | 高 | 類型用於一般開發、測試，以及需要高輸送量的生產工作負載。 |

## 組態

組態使用 Cuttlefish 格式，包含 RabbitMQ 代理程式的所有設定。您可以在建立任何代理程式之前建立組態。然後，您可以將組態套用至一或多個代理程式。

> **Important**
>
> 對組態進行變更，不會立即將變更套用至代理程式。若要套用變更，您必須等待下一個維護時段或重新啟動代理程式。如需詳細資訊，請參閱 Amazon MQ 代理程式組態生命週期。
> 您目前無法刪除組態。

如需建立、編輯和管理組態的詳細資訊，請參閱以下各節：

- Creating and applying broker configurations
- RabbitMQ Broker Configurations

若要追蹤您對組態做出的變更，您可以建立組態修訂。如需詳細資訊，請參閱 Creating and applying broker configurations。

### Attributes

代理程式組態具有多個屬性，例如：

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

如需組態屬性的完整清單，請參閱 Amazon MQ REST API 參考中的以下各節：

- REST 操作 ID：組態
- REST 操作 ID：組態

如需組態修訂屬性的完整清單，請參閱以下各節：

- REST 操作 ID：組態修訂
- REST 操作 ID：組態修訂

## 使用者

每個 AMQP 0-9-1 用戶端連線都有一個必須經過驗證的關聯使用者。每個用戶端連線也會以虛擬主機 (vhost) 為目標，使用者必須對其擁有一組許可。使用者可能有權設定、寫入，以及讀取虛擬主機中的佇列和交換。使用者登入資料和目標虛擬主機是在建立連線時指定。

當您第一次建立 Amazon MQ for RabbitMQ 代理程式時，Amazon MQ 會使用您提供的登入憑證來建立具有 `administrator` 標籤的 RabbitMQ 使用者。您可以接著透過 RabbitMQ 管理 API 或 RabbitMQ Web 主控台，新增及管理使用者。您也可以使用 RabbitMQ Web 主控台或管理 API 來設定或修改使用者許可和標籤。

> **Note**
>
> RabbitMQ 使用者不會透過 Amazon MQ Users (使用者) API 儲存或顯示。

> **Important**
>
> 適用於 RabbitMQ 的 Amazon MQ 不支援使用者名稱「訪客」，而且會在您建立新的代理程式時刪除預設來賓帳戶。Amazon MQ 也會定期刪除任何客戶建立的名為「訪客」的帳戶。

若要使用 RabbitMQ 管理 API 建立新使用者，請使用下列 API 端點和要求主體。以新的登入憑證取代*####*和*##*。

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

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

> **Important**
>
> - 請勿在代理程式使用者名稱中加入個人身分識別資訊 (PII) 或其他機密或敏感資訊。代理程式使用者名稱可AWS供其他服務存取，包括 CloudWatch 記錄檔。代理程式使用者名稱不適用於私有或敏感資料。
> - 如果您忘記在建立代理程式時設定的管理員密碼，則無法重設您的憑證。如果您已建立多個管理員，則可以使用其他管理員使用者登入，然後重設或重新建立您的憑證。如果您只有一個管理員使用者，則必須刪除該代理程式，並使用新憑證建立一個新的代理程式。我們建議在刪除代理程式之前先使用或備份訊息。

`tags` 索引鍵屬於強制性，而且是使用者以逗號分隔的標籤清單。Amazon MQ 支援 `administrator`、`management`、`monitoring` 及 `policymaker` 使用者標籤。

您可以使用下列 API 端點和要求主體來設定個別使用者的許可。以您的資訊取代 *`vhost`* 和 *`username`*。對於預設虛擬主機 `/`，使用 %2F。

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

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

> **Note**
>
> `configure`、`read` 及 `write` 索引鍵全都屬於強制性。

使用萬用字元 `.*` 值，此作業會將指定的虛擬主機中所有佇列的讀取、寫入和設定許可授予使用者。如需透過 RabbitMQ 管理 API 管理使用者的詳細資訊，請參閱 RabbitMQ 管理 HTTP API。

## 外掛程式

Amazon MQ for RabbitMQ 支援 RabbitMQ 管理外掛程式，其為管理 API 和 RabbitMQ Web 主控台提供支援。您可以使用 Web 主控台和管理 API 來建立和管理代理程式使用者和政策。

除了管理外掛程式，Amazon MQ for RabbitMQ 還支援下列外掛程式。

主題

- Shovel 外掛程式

- 聯合外掛程式
- 一致雜湊交換外掛程式

### Shovel 外掛程式

Amazon MQ 受管代理程式支援 RabbitMQ shovel，可讓您將訊息從一個代理程式執行個體移到另一個代理程式執行個體上的佇列和交換。您可以使用 shovel 連接鬆散耦合的代理程式，並將從具有較重訊息負載的節點分發出訊息。

Amazon MQ 受管 RabbitMQ 代理程式支援動態 shovel。動態 shovel 使用執行時間參數進行設定，並可經由用戶端連線以程式設計方式隨時啟動和停止。例如，使用 RabbitMQ 管理 API，您可以對以下 API 端點建立 PUT 要求以設定動態 shovel。在範例中，{vhost} 可由代理程式的虛擬主機名稱取代，而 {name} 可由新的動態 shovel 名稱取代。

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

在要求主體中，您必須指定佇列或交換，但不能同時指定兩者。下面的這個範例在 src-queue 中指定的本機佇列與 dest-queue 中定義的遠端佇列之間設定動態 shovel。同樣地，您可使用 src-exchange 和 dest-exchange 參數來設定兩個交換之間的 shovel。

```
{
  "value": {
    "src-protocol": "amqp091",
    "src-uri":  "amqp://localhost",
    "src-queue":  "source-queue-name",
    "dest-protocol": "amqp091",
    "dest-uri": "amqps://b-c8352341-ec91-4a78-ad9c-a43f23d325bb.mq.us-
west-2.amazonaws.com:5671",
    "dest-queue": "destination-queue-name"
  }
}
```

> **Important**
>
> 如果 shovel 目的地是私有代理程式，則無法在佇列或交換之間設定 shovel。您只能在公有代理程式中的佇列或交換之間設定 shovel，或在私有代理程式中的來源與公有代理程式中的目的地之間設定 shovel。

如需使用動態 shovel 的詳細資訊，請參閱 RabbitMQ 動態 shovel 外掛程式。

> **Note**
>
> Amazon MQ 不支援使用靜態 shovel。

### 聯合外掛程式

Amazon MQ 支援聯合交換和佇列。透過聯合，您可以在個別代理程式上的佇列、交換和消費者之間複寫訊息流程。聯合隊列和交易所使用 point-to-point 鏈接連接到其他經紀人中的對等。聯合交換時，預設會路由傳送訊息一次，聯合佇列可視消費者的需要多次移動訊息。

您可以使用聯合來允許下游代理程式從上游的交換或佇列中取用訊息。您可以使用 RabbitMQ Web 主控台或管理 API，啟用下游代理程式的聯合。

> **Important**
>
> 如果上游佇列或交換處於私有代理程式中，則無法設定聯合。您只能在公有代理程式中的佇列或交換之間，或者在公有代理程式中的上游佇列或交換與私有代理程式中的下游佇列或交換之間設定聯合。

例如，使用管理 API，您可以執行下列動作來設定同盟。

- 設定一或多個上游，以定義與其他節點的聯合連線。您可以使用 RabbitMQ Web 主控台或管理 API，定義聯合連線。使用管理 API，您可以使用以下要求主體來建立對 `/api/parameters/federation-upstream/%2f/my-upstream` 的 POST 要求。

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

- 設定政策，讓您的佇列或交換變得聯合。您可以使用 RabbitMQ Web 主控台或管理 API 來設定政策。使用管理 API，您可以使用以下要求主體來建立對 `/api/policies/%2f/federate-me` 的 POST 要求。

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

> **Note**
>
> 要求主體假設伺服器上的交換會命名為以 amq 開頭。使用常規表達式 ^amq\.，可確保所有名稱以 "amq" 開頭的交換已啟用聯合。RabbitMQ 伺服器上的交換可以不同的方式命名。

如需設定聯合外掛程式的詳細資訊，請參閱 RabbitMQ 聯合外掛程式。

一致雜湊交換外掛程式

預設情況下，Amazon MQ for RabbitMQ 支援一致雜湊交換類型外掛程式。一致雜湊交換會根據從訊息路由索引鍵計算的雜湊值，將訊息路由傳送到佇列。假設有合理平均的路由索引鍵，一致雜湊交換可以在佇列之間合理地平均分配訊息。

對於繫結至一致雜湊交換的佇列，繫結索引鍵 number-as-a-string 是決定每個佇列的繫結權數。具較高繫結權數的佇列會從其繫結的一致雜湊交換接收分配比例較高的訊息。在一致雜湊交換拓撲中，發行者可以直接將訊息發佈至交換，但是消費者必須明確設定為取用特定佇列中的訊息。

如需有關一致雜湊交換的詳細資訊，請參閱網站上的 RabbitMQ 一致雜湊交換類型。 GitHub

## 政策

您可以使用 Amazon MQ 建議的預設值套用自訂政策和限制。如果您已刪除建議的預設政策和限制，並想要予以重建，或者您已建立其他虛擬主機並想要將預設政策和限制套用至新的虛擬主機，則可使用下列步驟。

> **Important**
>
> 若要執行下列步驟，您必須擁有具管理員許可的 Amazon MQ for RabbitMQ 代理程式使用者。您可以使用第一次建立代理程式時建立的管理員使用者，或您之後可能建立的其他使用者。下表提供必要的管理員使用者標籤和許可作為規則表達式 (regexp) 模式。
>
> | 標籤 | 讀取 regexp | 設定 regexp | 寫入 regexp |
> | --- | --- | --- | --- |
> | `administrator` | `.*` | `.*` | `.*` |
>
> 如需建立 RabbitMQ 使用者及管理使用者標籤和許可的詳細資訊，請參閱 使用者。

使用 RabbitMQ Web 主控台套用預設政策和虛擬主機限制

1. 登入 Amazon MQ 主控台。
2. 在左側導覽窗格中選擇 Brokers (代理程式)。
3. 從代理程式清單中，選擇您要套用新政策的代理程式名稱。
4. 在代理程式詳細資訊頁面的 Connections (連線) 區段中，選擇 RabbitMQ web console (RabbitMQ Web 主控台) URL。RabbitMQ Web 主控台會在新的瀏覽器索引標籤或視窗中開啟。
5. 使用您的代理程式管理員使用者名稱和密碼登入 RabbitMQ Web 主控台。
6. 在 RabbitMQ Web 主控台的頁面頂端，選擇 Admin (管理員)。
7. 在 Admin (管理員) 頁面的右側導覽窗格中，選擇 Policies (政策)。
8. 在 Policies (政策) 頁面上，您可看到代理程式目前的 User policies (使用者政策) 清單。在 User policies (使用者政策) 下方，展開 Add / update a policy (新增/更新政策)。
9. 若要建立新的代理程式政策，請在 Add / update a policy (新增/更新政策) 之下，執行下列操作：

    a. 針對 Virtual host (虛擬主機)，從下拉式清單中選擇您要附加政策的虛擬主機名稱。若要選擇預設虛擬主機，請選擇 /。

    > **Note**
    >
    > 如果您尚未建立其他虛擬主機，Virtual host (虛擬主機) 選項不會顯示在 RabbitMQ 主控台中，而且政策只會套用至預設虛擬主機。

    b. 針對 Name (名稱)，輸入您的政策名稱，例如 **`policy-defaults`**。

    c. 針對 Pattern (模式)，輸入 regexp 模式 **`.*`**，以便政策符合代理程式上的所有佇列。

    d. 針對 Apply to (套用至)，從下拉式清單中選擇 Exchanges and queues (交換和佇列}。

    e. 對於 Priority (優先順序)，輸入大於套用至虛擬主機的所有其他策略的整數。您可以在任何指定的時間將一組政策定義套用至 RabbitMQ 佇列和交換。RabbitMQ 會選擇具有最高優先順序值的相符政策。如需政策優先順序及如何合併政策的詳細資訊，請參閱 RabbitMQ 伺服器文件中的政策。

    f. 針對 Definition (定義)，新增下列鍵值組：

    - **`queue-mode=lazy`**。從下拉式清單中選擇 String (字串)
    - **`overflow=reject-publish`**。從下拉式清單中選擇 String (字串)

> **Note**
>
> 不適用於單一執行個體的代理程式。

- **`max-length=`*`number-of-messages`***。 根據代*`number-of-messages`*理程式的執行個體大小和部署模式 (例如`mq.m5.large`叢集)，以 Amazon MQ 建議值**`8000000`**取代。從下拉式清單中選擇 Number (數字)。

> **Note**
>
> 不適用於單一執行個體的代理程式。

g. 選擇 Add / update policy (新增 / 更新政策)。

10. 確認新政策出現在 User policies (使用者政策) 清單中。

> **Note**
>
> 對於叢集代理程式，Amazon MQ 會自動套用 `ha-mode: all` 和 `ha-sync-mode: automatic` 政策定義。

11. 從右側導覽窗格中，選擇 Limits (限制)。
12. 在 Limits (限制) 頁面上，您可看到代理程式目前的 Virtual host limits (虛擬主機限制) 清單。在 Virtual host limits (虛擬主機限制) 下方，展開 Set / update a virtual host limit (設定 / 更新虛擬主機限制)。
13. 若要建立新的虛擬主機限制，請在 Set / update a virtual host limit (設定 / 更新虛擬主機限制) 之下，請執行下列動作：

    a. 針對 Virtual host (虛擬主機)，從下拉式清單中選擇您要附加政策的虛擬主機名稱。若要選擇預設虛擬主機，請選擇 /。

    b. 針對 Limit (限制)，從下拉式選項中選擇 max-connections。

    c. 針對 Value (值)，根據代理程式的執行個體大小和部署模式，例如 **`15000`** 適用於 `mq.m5.large` 叢集，輸入 Amazon MQ 建議的值。

    d. 選擇 Set / update limit (設定/更新限制)。

    e. 重複上述步驟，並針對 Limit (限制)，從下拉式選項中選擇 max-queues。

14. 確認新的限制會出現在 Virtual host limits (虛擬主機限制) 清單中。

使用 RabbitMQ 管理 API 套用預設政策和虛擬主機限制

1. 登入 Amazon MQ 主控台。
2. 在左側導覽窗格中選擇 Brokers (代理程式)。
3. 從代理程式清單中，選擇您要套用新政策的代理程式名稱。
4. 在代理程式頁面的 Connections (連線) 區段中，記下 RabbitMQ web console (RabbitMQ Web 主控台) URL。這是您在 HTTP 請求中使用的代理程式端點。
5. 開啟新的終端機或您所選的命令列視窗。
6. 若要建立新的代理程式政策，請輸入下列 `curl` 命令。此命令會假設預設 / 虛擬主機上的佇列，其編碼為 `%2F`。若要將策略套用至其他虛擬主機，請將 `%2F` 替換為虛擬主機的名稱。

   > **Note**
   >
   > 以您的管理員登入憑證取代*#####*和*##*。根據代*number-of-messages*理程式的執行個體大小和部署模式，以 Amazon MQ 建議值取代。以您的政策名稱取代 *policy-name*。以您先前記下的 URL 取代 *broker-endpoint*。

   ```
   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. 若要確認新政策已新增至代理程式的使用者政策，請輸入下列 `curl` 命令以列出所有代理程式政策。

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

8. 若要建立新的 `max-connections` 虛擬主機限制，請輸入下列 `curl` 命令。此命令會假設預設 / 虛擬主機上的佇列，其編碼為 `%2F`。若要將策略套用至其他虛擬主機，請將 `%2F` 替換為虛擬主機的名稱。

> **Note**
>
> 以您的管理員登入憑證取代*#####*和*##*。根據代理程式的執行個體大小和部署模式，使用 Amazon MQ 建議的值取代 *max-connections*。以您先前記下的 URL 取代代理程式端點。

```
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. 若要建立新的 max-queues 虛擬主機限制，請重複前一步，但如以下所示修改 curl 命令。

```
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. 若要確認新限制已新增至代理程式的虛擬主機限制，請輸入以下 curl 命令來列出所有代理程式虛擬主機限制。

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

## 代理程式架構

RabbitMQ 代理程式可以建立為單一執行個體代理程式或建立於叢集部署中。對於這兩種部署模式，Amazon MQ 會經由備援方式儲存資料，以提供高耐久性。

您可以使用 RabbitMQ 支援的任何程式設計語言，並為下列通訊協定啟用 TLS，以存取 RabbitMQ 代理程式：

- AMQP (0-9-1)

主題

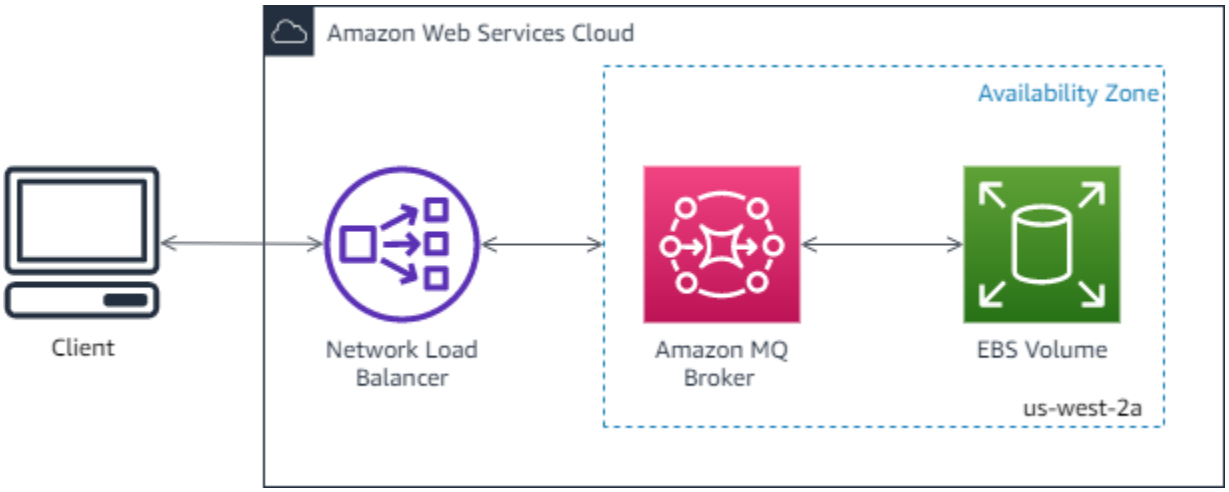
- 單一執行個體代理程式
- 高可用性的叢集部署

## 單一執行個體代理程式

單一執行個體代理程式是由 Network Load Balancer (NLB) 後面的一個可用區域中的一個代理程式組成。代理程式會與您的應用程式以及 Amazon EBS 儲存磁碟區進行通訊。Amazon EBS 提供針對低延遲和高輸送量最佳化的區塊層級儲存。

如果代理程式執行個體在維護時段期間或因為基礎 Amazon EC2 硬體故障而被取代，則使用 Network Load Balancer 可確保 Amazon MQ for RabbitMQ 代理程式端點保持不變。Network Load Balancer 可讓您的應用程式和使用者繼續使用相同的端點來連接至代理程式。

下圖說明 Amazon MQ for RabbitMQ 單一執行個體代理程式。




## 高可用性的叢集部署

叢集部署是 Network Load Balancer 後面的三個 RabbitMQ 代理程式節點的邏輯分組，每個節點共用使用者、佇列，以及跨多個可用區域 (AZ) 的分散式狀態。

在叢集部署中，Amazon MQ 會自動管理代理程式政策，以啟用跨所有節點的傳統鏡像，進而確保高可用性 (HA)。每個鏡像佇列都包含一個主要節點和一或多個鏡像。每個佇列都有自己的主要節點。指定佇列的所有作業都會先套用在佇列的主要節點上，然後傳播到鏡像。Amazon MQ 會建立預設系統政策，以將 `ha-mode` 設為 `all` 和將 `ha-sync-mode` 設為 `automatic`。這可確保資料會跨不同可用區域複寫到叢集中的所有節點，以獲得更佳的耐久性。

> **Note**
>
> 在維護時段，叢集的所有維護會一次執行一個節點，而且隨時保留至少兩個執行中的節點。每次關閉節點時，該節點的用戶端連線都會被切斷，而且需要重建。您必須確保用戶端程式碼設計為自動重新連線到叢集。如需連線復原的詳細資訊，請參閱 the section called “從網路故障中自動復原”。

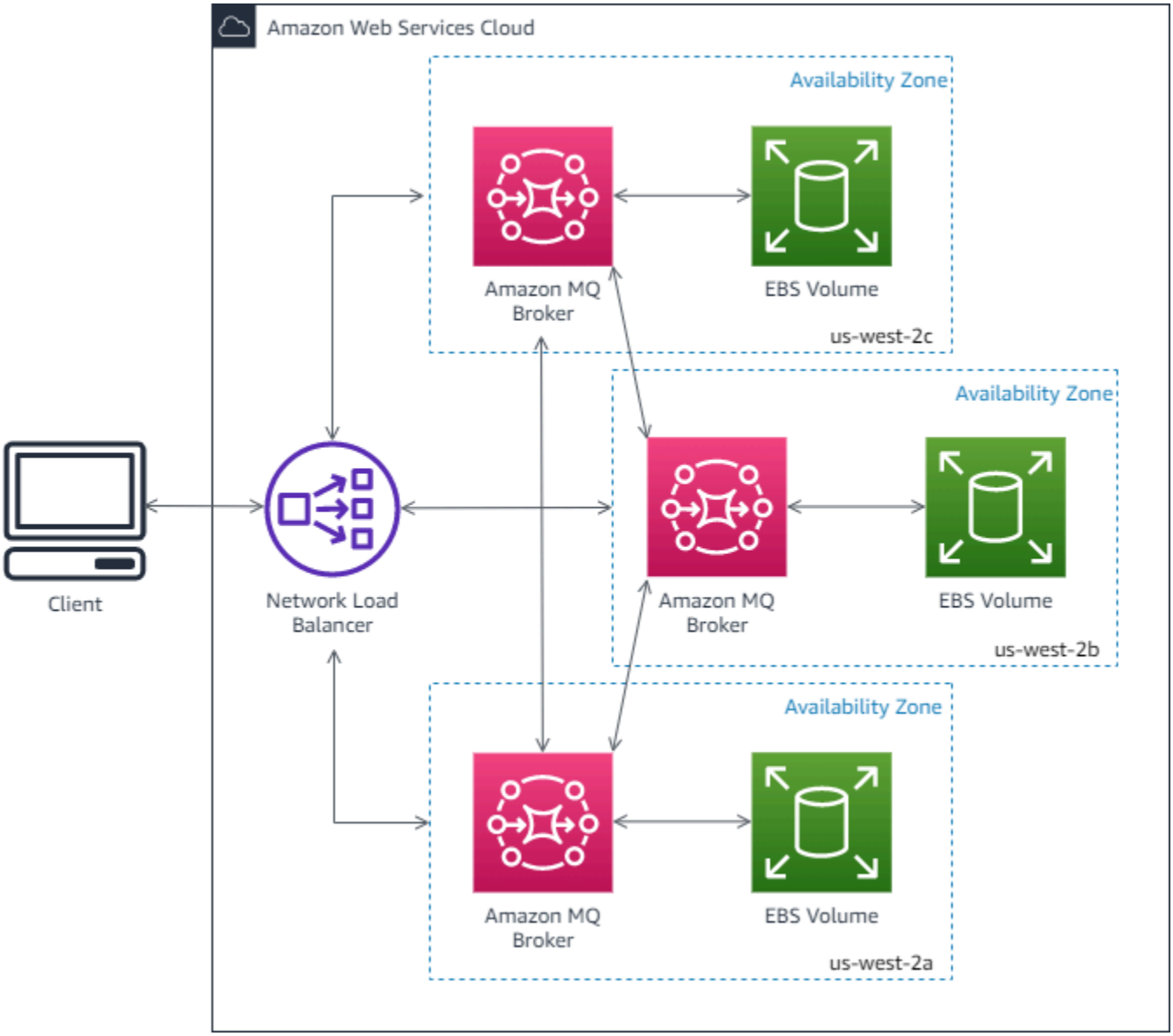
由於 Amazon MQ 設定 `ha-sync-mode: automatic`，在維護期間，當每個節點重新加入叢集時，佇列就會同步處理。佇列同步處理會封鎖所有其他佇列操作。您可以讓佇列保持簡短，以減輕在維護時段對佇列同步處理的影響。

不應刪除預設政策。如果您刪除此政策，Amazon MQ 將會予以自動重建。Amazon MQ 也會確保 HA 屬性套用至您在叢集式代理程式上建立的所有其他政策。如果您新增不含 HA 屬性的政策，Amazon MQ 會為您新增這些屬性。如果您新增具不同高可用性屬性的政策，Amazon MQ 將會取代這些屬性。如需傳統鏡像的詳細資訊，請參閱傳統鏡像佇列。

**Important**

Amazon MQ 不支援仲裁佇列。啟用仲裁佇列功能旗標並建立仲裁佇列會導致資料遺失。

下圖說明 RabbitMQ 叢集代理程式部署，其中有三個節點位於三個可用區域 (AZ)，每個節點都有自己的 Amazon EBS 磁碟區和共用狀態。Amazon EBS 提供針對低延遲和高輸送量最佳化的區塊層級儲存。




# Amazon MQ for RabbitMQ 代理程式組態

組態使用 Cuttlefish 格式，包含 RabbitMQ 代理程式的所有設定。您可以在建立任何代理程式之前建立組態。然後，您可以將組態套用至一或多個代理程式。

**主題**

- 建立、編輯和套用 RabbitMQ 代理程式組態
- RabbitMQ 組態政策

# 建立、編輯和套用 RabbitMQ 代理程式組態

組態使用 Cuttlefish 格式，包含 RabbitMQ 代理程式的所有設定。您可以在建立任何代理程式之前建立組態。然後，您可以將組態套用至一或多個代理程式。

如需詳細資訊，請參閱下列內容：

- 組態
- Amazon MQ 代理程式組態生命週期

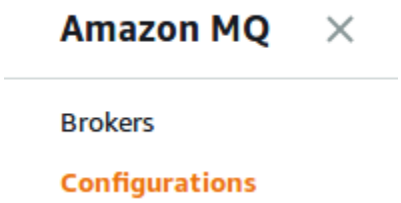
以下範例顯示如何使用 AWS Management Console 建立和套用 RabbitMQ 代理程式組態。

主題

- 建立新組態
- 建立新的組態修訂
- 將組態修訂套用至您的代理程式
- 編輯組態修訂

建立新組態

1. 登入 Amazon MQ 主控台。
2. 在左邊，展開導覽面板並選擇 Configurations (組態)。




3. 在 Configurations (組態) 頁面上，選擇 Create configuration (建立組態)。
4. 在 Create configuration (建立組態) 頁面的 Details (詳細資訊) 區段中，輸入 Configuration name (組態名稱) (例如 `MyConfiguration`)，然後選取 Broker engine (代理程式引擎) 版本。

   若要深入了解 Amazon MQ for RabbitMQ 支援的 RabbitMQ 引擎版本，請參閱 the section called "版本管理"。
5. 選擇建立組態。

建立新的組態修訂

1. 從組態清單中選擇*MyConfiguration*。

   > **Note**
   >
   > 當 Amazon MQ 建立組態時，一律為您建立第一個組態修訂。

   在*MyConfiguration*頁面上，會顯示新組態修訂版本使用的代理程式引擎類型和版本 (例如 RabbitMQ 3.xx .xx)。

2. 組態詳細資訊標籤上會顯示組態修訂編號、描述及 Cuttlefish 格式的代理程式組態。

   > **Note**
   >
   > 編輯目前的組態會建立新的組態版本。

3. 選擇編輯組態，然後變更 Cuttlefish 組態。
4. 選擇儲存。

   Save revision (儲存修訂) 對話方塊隨即顯示。

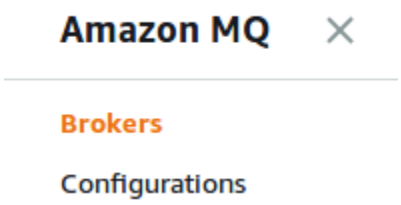
5. (選用) 輸入 `A description of the changes in this revision`。
6. 選擇 Save (儲存)。

   隨即儲存組態的新修訂版。

   > **Important**
   >
   > 對組態進行變更，不會立即將變更套用至代理程式。若要套用變更，您必須等待下一個維護時段或重新啟動代理程式。如需詳細資訊，請參閱 Amazon MQ 代理程式組態生命週期。
   > 您目前無法刪除組態。

將組態修訂套用至您的代理程式

1. 在左邊，展開導覽面板並選擇 Brokers (代理程式)。




2. 從經紀人清單中，選取您的經紀人 (例如，MyBroker)，然後選擇編輯。
3. 在「編輯」*MyBroker*頁面的「組態」段落中，選取「組態」和「修訂版本」，然後選擇「排程修改」。
4. 在 Schedule broker modifications (排定代理程式修改) 部分，選擇套用修改的時機是 During the next scheduled maintenance window (下一個排程的維護時段) 或 Immediately (立即)。

> **Important**
>
> 您的代理程式會在重新啟動時離線。

5. 選擇套用。

   組態修訂會在指定時間套用至代理程式。

### 編輯組態修訂

1. 登入 Amazon MQ 主控台。
2. 從經紀人清單中，選取您的經紀人 (例如，MyBroker)，然後選擇編輯。
3. 在***MyBroker***頁面上，選擇 [編輯]。
4. 在「編輯」*MyBroker*頁面的「組態」區段中，選取「組態」和「修訂版本」，然後選擇「編輯」。

> **Note**
>
> 除非您在建立代理程式時選取組態，否則一律會在 Amazon MQ 在建立代理程式時為您建立第一個組態修訂版。

   在***MyBroker***頁面上，會顯示組態使用的代理程式引擎類型和版本 (例如 RabbitMQ 3.xx .xx)。

5. 組態詳細資訊標籤上會顯示組態修訂編號、描述及 Cuttlefish 格式的代理程式組態。

> **Note**
>
> 編輯目前的組態會建立新的組態版本。

6. 選擇編輯組態，然後變更 Cuttlefish 組態。
7. 選擇儲存。

   Save revision (儲存修訂) 對話方塊隨即顯示。
8. (選用) 輸入 `A description of the changes in this revision`。
9. 選擇 Save (儲存)。

   隨即儲存組態的新修訂版。

> **Important**
>
> 對組態進行變更，不會立即將變更套用至代理程式。若要套用變更，您必須等待下一個維護時段或重新啟動代理程式。如需詳細資訊，請參閱 Amazon MQ 代理程式組態生命週期。
> 您目前無法刪除組態。

## RabbitMQ 組態政策

Amazon MQ for RabbitMQ 現在支援建立和套用組態至您的 RabbitMQ 代理程式。每部虛擬主機上的預設操作員政策都具有下列建議的 HA 屬性：

```
name: default_operator_policy_AWS_managed
pattern: .*
apply-to: all
priority: 0
definition: {
ha-mode: all
ha-sync-mode: automatic
}
```

根據預設，無法透過 AWS Management Console 或 Management API 對操作員政策進行變更。您可以將下面這行新增至代理程式組態來啟用變更：

```
management.restrictions.operator_policy_changes.disabled=false
```

如果您進行此變更，我們強烈建議您將 HA 屬性納入您自己的操作員政策中。如需新增組態至代理程式的詳細資訊，請參閱 Creating and applying broker configurations。

## 管理 Amazon MQ for RabbitMQ 引擎版本

RabbitMQ 會根據語義版本控制規格將版本號碼組織為 `X.Y.Z`。在 Amazon MQ for RabbitMQ 實作中，`X.Y` 表示主要版本，而 `Z` 代表次要版本號碼。如果主要版本號碼發生變更，Amazon MQ 會將版本變更視為主要版本變更。例如，從 3.8 版升級至 3.9 版被視為主要版本升級。如果只有次要版本號碼有所改變，則視為次要版本變更。例如，從 3.8.23 版升級至 3.8.26 版被視為次要版本升級。

當您建立新的 Amazon MQ for RabbitMQ 時，您可以指定任何支援的 RabbitMQ 引擎版本。如果您使用建立代理程式，Amazon MQ 會自動預設為最新的引擎版本號碼。 AWS Management Console 如果您使用 AWS CLI 或 Amazon MQ API 建立代理程式，則需要提供引擎版本號碼。如果您不提供版本號碼，則操作會導致例外狀況。如需進一步了解，請參閱 AWS CLI 命令參考中的 `create-broker` 和 Amazon MQ REST API 參考中的 `CreateBroker`。

> **Important**
>
> Amazon MQ 不支援仲裁佇列或串流。啟用這些功能旗標並建立仲裁佇列或串流會導致資料遺失。

> **Important**
>
> - Amazon MQ 不支援在 RabbitMQ 3.9 中推出的 JSON 中使用結構化日誌記錄。
> - Amazon MQ for RabbitMQ 建議新的代理程式使用最新支援的次要版本。
> - RabbitMQ 只允許增量版本更新 (例如：3.9.x 至 3.10.x)。更新時，您無法略過主要版本 (例如：3.8.x 至 3.10.x)。

Amazon MQ for RabbitMQ 目前支援下列引擎版本：

| 主要版本 | 次要版本 |
| --- | --- |
| RabbitMQ 3.11 | • 11 月 28 日 (建議使用) |
| RabbitMQ 3.10 | • 3.10.25 |
| RabbitMQ 3.9 | • 3.9.27 |
| RabbitMQ 3.8 | • 3.8.34 |

下列次要版本仍可用於現有代理程式，但不建議新的代理程式使用。

現有次要版本

| 主要版本 | 次要版本 |
| --- | --- |
| RabbitMQ 3.11 | • 3.11.20<br>• 3.11.16 |
| RabbitMQ 3.10 | • 3.10.20<br>• 3.10.10 |
| RabbitMQ 3.9 | • 3.9.24<br>• 3.9.16<br>• 3.9.13 |
| RabbitMQ 3.8 | • 3.8.30<br>• 3.8.27<br>• 3.8.26<br>• 3.8.23<br>• 3.8.22<br>• 3.8.17<br>• 3.8.6 |

### 主要和次要版本升級

透過 Amazon MQ，即可控制將代理程式升級至新版本的時機。若已啟用自動次要版本升級，Amazon MQ 將自動將您的代理引擎升級到由 Amazon MQ 發行並支援的新次要版本。

若要執行主要版本升級，您必須手動升級代理程式的引擎版本號碼。次要和主要版本升級可以在您排定的維護時段內，與其他代理程式修補作業同時發生。若您選擇不要自動次要版本升級，可以按照與主要版本相同的程序，將代理程式手動升級至新支援的次要版本。

如需更新代理程式偏好設定以啟用或停用次要版本升級，以及手動升級代理程式的詳細資訊，請參閱 the section called “升級引擎版本”。

單一執行個體代理程式會在重新啟動時離線。對於叢集代理程式，鏡像佇列必須在重新開機期間同步。佇列越長，佇列同步處理程序可能需要更長的時間。在佇列同步處理期間，取用者和生產者無法使用佇列。佇列同步處理程序完成後，Broker 會再次變為可用。為了將影響降到最低，我們建議您在低流量時間進行升級。如需版本升級最佳作法的詳細資訊，請參閱Amazon MQ for RabbitMQ 最佳實踐。

### 列出支援的引擎版本

您可以使用`describe-broker-instance-options` AWS CLI 指令列出所有受支援的次要和主要引擎版本。

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

若要依照引擎和執行個體類型篩選結果，請使用 `--engine-type` 和 `--host-instance-type` 選項，如下所示。

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

例如，若要篩選 RabbitMQ 和 `mq.m5.large` 執行個體類型的結果，請將 *`engine-type`* 取代為 `RABBITMQ` 以及將 *`instance-type`* 取代為 `mq.m5.large`。

## RabbitMQ 教學

以下教學說明如何在 Amazon MQ 上設定和使用 RabbitMQ。若要進一步了解如何以各種程式設計語言 (例如 Node.js、Python、.NET 等) 使用支援的用戶端程式庫，請參閱《RabbitMQ 入門指南》中的 RabbitMQ 教學。

主題



# 編輯代理程式偏好設定

您可以編輯代理程式偏好設定，例如使用 AWS Management Console 啟用或停用 CloudWatch Logs。

## 編輯 RabbitMQ 代理程式選項

1. 登入 Amazon MQ 主控台。
2. 從代理程式清單，選取您的代理程式 (例如 MyBroker)，然後選擇 Edit (編輯)。
3. 在 Edit ***MyBroker*** (編輯 MyBroker) 頁面上，於 Specifications (規格) 區段中，選取 Broker engine version (代理程式引擎版本) 或Broker Instance type (代理程式執行個體類型)。
4. 在 CloudWatch Logs 區段中，按一下切換按鈕以啟用或停用一般日誌。不需要執行其他步驟。

   > **Note**
   >
   > - 對於 RabbitMQ 代理程式，Amazon MQ 會自動使用服務連結的角色 (SLR) 將一般日誌發佈到 CloudWatch。如需詳細資訊，請參閱 the section called “使用服務連結角色”。
   > - Amazon MQ 不支援 RabbitMQ 代理程式的稽核記錄。

5. 在 Maintenance (維護) 區段中，設定代理程式的維護排程：

   若要在 AWS 發佈新版本時，將代理程式升級至該版本，請選擇 Enable automatic minor version upgrades (啟用自動次要版本升級)。自動升級會發生於由星期幾、一天中的時間 (24 小時制) 和時區 (預設為 UTC) 所定義的維護時段期間。
6. 選擇 Schedule Modifications (排程修改)。

   > **Note**
   >
   > 如果您只選擇 Enable automatic minor version upgrades (啟用自動次要版本升級)，按鈕會變更為 Save (儲存)，因為無需重新啟動代理程式。

會在指定時間將您的偏好設定套用至代理程式。

# 將 Python Pika 與 Amazon MQ for RabbitMQ 搭配使用

以下教學課程說明如何設定 Python Pika 用戶端，同時設定 TLS 以便連線到 Amazon MQ for RabbitMQ 代理程式。Pika 是 RabbitMQ 之 AMQP 0-9-1 協定的 Python 實作。本教學課程將引導您完成安裝 Pika、宣告佇列、設定發行者將訊息傳送至代理程式的預設交換，以及設定從佇列接收訊息的取用者。

主題



## 先決條件

若要完成本教學課程的步驟，您需要以下先決條件：

- 一個 Amazon MQ for RabbitMQ 代理程式。如需詳細資訊，請參閱建立 Amazon MQ for RabbitMQ 代理程式。
- 您的作業系統應安裝 Python 3。
- 使用 Python `pip` 安裝 Pika。若要安裝 Pika，請開啟新的終端機視窗並執行以下命令。

```
$ python3 -m pip install pika
```

## 許可

在本教學課程中，您至少需要一個 Amazon MQ for RabbitMQ 代理程式使用者，其具有寫入和讀取虛擬主機的許可。下表將必要的最低許可描述為規則表達式 (regexp) 模式。

| Tags (標籤) | 設定 regexp | 寫入 regexp | 讀取 regexp |
| --- | --- | --- | --- |
| none | | .* | .* |

列出的使用者許可僅為使用者提供讀取和寫入許可，而不會授予對管理外掛程式的存取權，以便在代理程式上執行管理操作。您可以提供限制使用者存取指定佇列的規則運算式模式，進一步限制許可。例如，如果您將讀取規則表達式模式變更為 `^[hello world].*`，則使用者將僅擁有從以 `hello world` 開頭之佇列中進行讀取的許可。

如需建立 RabbitMQ 使用者及管理使用者標籤和許可的詳細資訊，請參閱 使用者。

## 步驟一：建立基本 Python Pika 用戶端

若要建立 Python Pika 用戶端基本類別，以定義建構函數並在與 Amazon MQ for RabbitMQ 代理程式互動時提供 TLS 組態所需的 SSL 內容，請執行下列動作。

1. 開啟新終端機視窗，為您的專案建立新目錄，並導覽至該目錄。

```
$ mkdir pika-tutorial
$ cd pika-tutorial
```

2. 建立新檔案 `basicClient.py`，其包含下列 Python 程式碼。

```
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()
```

您現在可以為繼承自 BasicPikaClient 的發行者和取用者定義其他類別。

## 步驟二：建立發行者並傳送訊息

若要建立宣告佇列並傳送單一訊息的發行者，請執行下列動作。

1. 複製下列程式碼範例的內容，並在上一個步驟建立的相同目錄中本機儲存為 publisher.py。

```
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 類別繼承自 BasicPikaClient 並實作其他方法來宣告佇列、向佇列傳送訊息以及關閉連線。程式碼範例會將訊息路由傳送至預設交換，且路由金鑰等於佇列名稱。

2. 在 `if __name__ == "__main__":` 下方，請用下列資訊取代傳遞給 BasicMessageSender 建構函數陳述式的參數。

   - **<broker-id>** – Amazon MQ 針對代理程式產生的唯一 ID。您可以從代理程式 ARN 解析 ID。例如，假定是以下 ARN `arn:aws:mq:us-east-2:123456789012:broker:MyBroker:b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9`，代理程式 ID 會是 `b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9`。
   - **<username>** – 具有足夠許可可將訊息寫入代理程式的代理程式使用者的使用者名稱。
   - **<password>** – 具有足夠許可可將訊息寫入代理程式的代理程式使用者的密碼。
   - **<region>** – AWS 區域，您在其中建立了 Amazon MQ for RabbitMQ 代理程式。例如 `us-west-2`。

3. 在您建立 `publisher.py` 的相同目錄中執行下列命令。

```
$ python3 publisher.py
```

如果程式碼執行成功，則您將在終端機視窗中看到下列輸出。

```
Trying to declare queue(hello world queue)...
Sent message. Exchange: , Routing Key: hello world queue, Body: b'Hello World!'
```

## 步驟三：建立取用者並接收訊息

若要建立從佇列接收單一訊息的取用者，請執行下列動作。

1. 複製下列程式碼範例的內容，並在相同目錄中本機儲存為 `consumer.py`。

```
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()
```

與您在上一個步驟建立的發行者類似，BasicMessageReciever 繼承自 BasicPikaClient 並實作其他方法來接收單一訊息並關閉連線。

2. 在 `if __name__ == "__main__":` 下方，請用您的資訊取代傳遞給 BasicMessageReciever 建構函數的參數。
3. 在您的專案目錄中，執行下列命令。

```
$ python3 consumer.py
```

如果程式碼成功執行，則您會看到訊息內文以及包括路由金鑰在內的標頭，它們會顯示在終端機視窗中。

```
<Basic.GetOk(['delivery_tag=1', 'exchange=', 'message_count=0',
 'redelivered=False', 'routing_key=hello world queue'])> <BasicProperties> b'Hello
 World!'
```

## 步驟四：(選用) 設定事件迴圈並取用訊息

若要取用佇列中的多條訊息，請使用 Pika 的 basic_consume 方法和回呼函數，如下所示

1. 在 consumer.py 中，請將下列方法定義新增至 BasicMessageReceiver 類別。

```
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，在 `if __name__ == "__main__":` 下方，叫用您在上一步驟中定義的 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. 再次執行 `consumer.py`，如果成功，佇列訊息會顯示在終端機視窗中。

```
[*] Waiting for messages. To exit press CTRL+C
[x] Received b'Hello World!'
[x] Received b'Hello World!'
...
```

### 後續步驟？

- 如需其他支援 RabbitMQ 用戶端程式庫的詳細資訊，請參閱 RabbitMQ 網站上的 RabbitMQ 用戶端文件。

## 解決 RabbitMQ 暫停的佇列同步

在 Amazon MQ for RabbitMQ 叢集部署中，發佈至每個佇列的訊息會跨三個代理程式節點進行複寫。此複寫 (稱為鏡像) 為 RabbitMQ 代理程式提供高可用性 (HA)。叢集部署中的佇列由一個節點上的主要複本和一或多個鏡像所組成。每個套用至鏡像佇列的作業 (包括排入佇列的訊息) 都會先套用至主要佇列，然後跨鏡像進行複寫。

例如，考量跨三個節點複寫的鏡像佇列：主要節點 (`main`) 和兩個鏡像 (`mirror-1` 和 `mirror-2`)。如果此鏡像佇列中的所有訊息都會成功傳播到所有鏡像，則會同步處理此佇列。如果節點 (`mirror-1`) 在一段時間內變得無法使用，佇列仍然可以運作，而且可以繼續將訊息排入佇列。不過，對於要同步處理的佇列，則在 `mirror-1` 無法使用時發佈至 `main` 的訊息必須複寫到 `mirror-1`。

如需鏡像的詳細資訊，請參閱 RabbitMQ 網站上的傳統鏡像佇列。

維護與佇列同步

在維護時段期間，Amazon MQ 會一次執行一個節點的所有維護工作，以確保代理程式維持運作狀態。因此，佇列可能需要在每個節點繼續操作時進行同步處理。在同步期間，需要複寫到鏡像的訊息會從對

應的 Amazon Elastic Block Store (Amazon EBS) 磁碟區載入記憶體中，以便分批處理。分批處理訊息可讓佇列更快速地同步處理。

如果佇列保持簡短且訊息很小，則佇列會成功同步處理並繼續如預期般操作。不過，如果批次中的資料量接近節點的記憶體限制，節點就會引發高記憶體警示，暫停佇列同步。您可比較 `RabbitMemUsed` 與 `RabbitMqMemLimit` CloudWatch 中的代理程式節點指標，以確認記憶體使用量。直到取用或刪除訊息或減少批次中的訊息數目，才能完成同步處理。

> **Note**
>
> 減少佇列同步處理批次大小可能會導致較高的複寫交易數目。

若要解決暫停的佇列同步處理，請遵循本教學中的步驟，其中示範如何套用 `ha-sync-batch-size` 政策並重新啟動佇列同步處理。

主題

- 先決條件
- 步驟 1：套用 ha-sync-batch-size 政策
- 步驟 2：重新啟動佇列同步
- 後續步驟
- 相關資源

## 先決條件

在本教學中，您必須有具備管理員許可的 Amazon MQ for RabbitMQ 代理程式使用者。您可以使用第一次建立代理程式時建立的管理員使用者，或您之後可能建立的其他使用者。下表提供必要的管理員使用者標籤和許可作為規則表達式 (regexp) 模式。

| Tags (標籤) | 讀取 regexp | 設定 regexp | 寫入 regexp |
|---|---|---|---|
| `administrator` | `.*` | `.*` | `.*` |

如需建立 RabbitMQ 使用者及管理使用者標籤和許可的詳細資訊，請參閱 使用者。

## 步驟 1：套用 **ha-sync-batch-size** 政策

下列程序示範如何新增適用於代理程式上建立之所有佇列的政策。您可以使用 RabbitMQ Web 主控台或 RabbitMQ 管理 API。如需詳細資訊，請參閱 RabbitMQ 網站上的管理外掛程式。

使用 RabbitMQ Web 主控台套用 **ha-sync-batch-size** 政策

1. 登入 Amazon MQ 主控台。
2. 在左側導覽窗格中選擇 Brokers (代理程式)。
3. 從代理程式清單中，選擇您要套用新政策的代理程式名稱。
4. 在代理程式頁面的 Connections (連線) 區段中，選擇 RabbitMQ web console (RabbitMQ Web 主控台) URL。RabbitMQ Web 主控台會在新的瀏覽器索引標籤或視窗中開啟。
5. 使用您的代理程式管理員登入憑證登入 RabbitMQ Web 主控台。
6. 在 RabbitMQ Web 主控台的頁面頂端，選擇 Admin (管理員)。
7. 在 Admin (管理員) 頁面的右側導覽窗格中，選擇 Policies (政策)。
8. 在 Policies (政策) 頁面上，您可看到代理程式目前的 User policies (使用者政策) 清單。在 User policies (使用者政策) 下方，展開 Add / update a policy (新增/更新政策)。

   > **Note**
   >
   > 根據預設，使用名為 `ha-all-AWS-OWNED-DO-NOT-DELETE` 的初始代理程式政策建立 Amazon MQ for RabbitMQ 叢集。Amazon MQ 會管理此政策，以確保代理程式上的每個佇列都會複寫到所有三個節點，並自動同步佇列。

9. 若要建立新的代理程式政策，請在 Add / update a policy (新增/更新政策) 之下，執行下列操作：

   a. 針對 Name (名稱)，輸入您的政策名稱，例如 **batch-size-policy**。

   b. 針對 Pattern (模式)，輸入 regexp 模式 **.***，以便政策符合代理程式上的所有佇列。

   c. 針對 Apply to (套用至)，從下拉式清單中選擇 Exchanges and queues (交換和佇列}。

   d. 針對 Priority (優先順序)，輸入大於套用至虛擬主機的所有其他策略的整數。您可以在任何指定的時間將一組政策定義套用至 RabbitMQ 佇列和交換。RabbitMQ 會選擇具有最高優先順序值的相符政策。如需政策優先順序及如何合併政策的詳細資訊，請參閱 RabbitMQ 伺服器文件中的政策。

   e. 針對 Definition (定義)，新增下列鍵值組：

      - **ha-sync-batch-size**=*100*。從下拉式清單中選擇 Number (數字)。

> **Note**
>
> 您可能需要根據佇列中未同步的訊息數目和大小，調整和校準 `ha-sync-batch-size` 的值。

- **`ha-mode=all`**。從下拉式清單中選擇 String (字串)。

> **Important**
>
> 所有 HA 相關政策都需要 `ha-mode` 定義。省略它會導致驗證失敗。

- **`ha-sync-mode=automatic`**。從下拉式清單中選擇 String (字串)。

> **Note**
>
> 所有自訂政策都需要 `ha-sync-mode` 定義。若已省略，Amazon MQ 會自動附加定義。

f. 選擇 Add / update policy (新增 / 更新政策)。

10. 確認新政策出現在 User policies (使用者政策) 清單中。

使用 RabbitMQ 管理 API 套用 **`ha-sync-batch-size`** 政策

1. 登入 Amazon MQ 主控台。
2. 在左側導覽窗格中選擇 Brokers (代理程式)。
3. 從代理程式清單中，選擇您要套用新政策的代理程式名稱。
4. 在代理程式頁面的 Connections (連線) 區段中，記下 RabbitMQ web console (RabbitMQ Web 主控台) URL。這是您在 HTTP 請求中使用的代理程式端點。
5. 開啟新的終端機或您所選的命令列視窗。
6. 若要建立新的代理程式政策，請輸入下列 `curl` 命令。此命令會假設預設 / 虛擬主機上的佇列，其編碼為 `%2F`。

> **Note**
>
> 將*#####*和*##*取代為您的代理管理員登入憑證。您可能需要根據佇列中未同步的訊息數目和大小，調整和校準 ha-sync-batch-size 的值 (*100*)。以您先前記下的 URL 取代代理程式端點。

```
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. 若要確認新政策已新增至代理程式的使用者政策，請輸入下列 curl 命令以列出所有代理程式政策。

```
curl -i -u username:password https://b-589c045f-f8ln-4ab0-a89c-co62e1c32ef8.mq.us-
west-2.amazonaws.com/api/policies
```

## 步驟 2：重新啟動佇列同步

在將新的 ha-sync-batch-size 政策套用至代理程式之後，請重新啟動佇列同步處理。

使用 RabbitMQ Web 主控台重新啟動佇列同步

> **Note**
>
> 若要開啟 RabbitMQ Web 主控台，請參閱本教學的步驟 1 中的先前指示。

1. 在 RabbitMQ Web 主控台的頁面頂端，選擇 Queues (佇列)。
2. 在 Queues (佇列) 頁面的 All queues (所有佇列) 之下，找出您已暫停的佇列。在 Features (功能) 欄中，您的佇列應會列出您建立的新政策名稱 (例如 batch-size-policy)。
3. 若要以縮減的批次大小重新啟動同步程序，請選擇 Restart sync (重新啟動同步)。

> **Note**
>
> 如果同步處理暫停且無法順利完成，請嘗試縮減 `ha-sync-batch-size` 值，然後再次重新啟動佇列同步處理。

### 後續步驟

- 佇列成功同步後，您可檢視 Amazon CloudWatch 指標 `RabbitMQMemUsed` 以監控 RabbitMQ 節點使用的記憶體數量。您也可以檢視 `RabbitMQMemLimit` 指標來監控節點的記憶體限制。如需詳細資訊，請參閱 存取 Amazon MQ 的 CloudWatch 指標 及 記錄和監控 Amazon MQ for RabbitMQ 代理程式。
- 若要避免暫停的佇列同步處理，建議將佇列保持簡短並處理訊息。對於訊息大小較大的工作負載，我們也建議將代理程式執行個體類型升級為具有更多記憶體的較大執行個體大小。如需代理程式執行個體類型和編輯代理程式偏好設定的詳細資訊，請參閱 Amazon MQ for RabbitMQ 執行個體類型 和 編輯代理程式偏好設定。
- 當您建立新的 Amazon MQ for RabbitMQ 代理程式時，Amazon MQ 會套用一組預設政策和虛擬主機限制，以最佳化代理程式效能。如果您的代理程式沒有建議的預設政策和限制，我們建議您自行建立。如需建立預設政策和虛擬主機限制的詳細資訊，請參閱 the section called “代理程式預設值”。

### 相關資源

- UpdateBrokerInput – 使用 Amazon MQ API ，可利用此代理程式屬性更新代理程式執行個體類型。
- 參數和政策 (RabbitMQ 伺服器文件) – 進一步了解 RabbitMQ 網站上的 RabbitMQ 參數和政策。
- RabbitMQ 管理 HTTP API – 進一步了解 RabbitMQ 管理 API。

## Amazon MQ for RabbitMQ 最佳實踐

使用本節作為參考，快速找到在 Amazon MQ 上使用 RabbitMQ 代理程式時用於提升效能並降低輸送量成本的建議。

> **Important**
>
> Amazon MQ 不支援仲裁佇列。啟用仲裁佇列功能旗標並建立仲裁佇列會導致資料遺失。

> **Important**
>
> 目前，Amazon MQ 不支援串流，也無法在 RabbitMQ 3.9.x 中推出的 JSON 中使用結構化日誌記錄。

> **Important**
>
> 適用於 RabbitMQ 的 Amazon MQ 不支援使用者名稱「訪客」，而且會在您建立新的代理程式時刪除預設來賓帳戶。Amazon MQ 也會定期刪除任何客戶建立的名為「訪客」的帳戶。

主題



## 啟用延遲佇列

如果您正在處理非常長的佇列，這些佇列會處理大量訊息，啟用延遲佇列可以改善代理程式的整體效能。

RabbitMQ 的預設行為是快取記憶體中的訊息，而只有在代理程式需要更多可用記憶體時，將這些訊息移到磁碟。將訊息從記憶體移至磁碟的這個程序可能需要一些時間，並使佇列停止處理訊息。啟用延遲佇列可能會對加速將訊息移至磁碟的程序造成重大影響，因為延遲佇列會儘快將訊息儲存至磁碟，從而減少記憶體中快取的訊息。

您可在宣告時設定 `queue.declare` 引數，或透過 RabbitMQ 管理主控台設定政策，以啟用延遲佇列。以下範例示範如何使用 RabbitMQ Java 用戶端程式庫宣告延遲佇列。

```
Map<String, Object> args = new HashMap<String, Object>();
```

```
args.put("x-queue-mode", "lazy");
channel.queueDeclare("myqueue", false, false, false, args);
```

> **Note**
>
> 啟用延遲佇列可以增加磁碟 I/O 操作。

## 使用持續且耐久的佇列

持續性訊息有助於防止代理程式當機或重新啟動時資料遺失。持續性訊息會在送達磁碟時立即寫入。然而，與延遲佇列不同，除非代理程式需要更多記憶體，否則持續性訊息會在記憶體和磁碟中快取。在需要更多記憶體的情況下，訊息由管理將訊息儲存到磁碟的 RabbitMQ 代理程式機制從記憶體中刪除，通常稱為持續性層。

若要啟用訊息持續性，您可以將佇列宣告為 `durable`，並將訊息傳遞模式設定為 `persistent`。以下範例示範如何使用 RabbitMQ Java 用戶端程式庫宣告耐久的佇列。

```
boolean durable = true;
channel.queueDeclare("my_queue", durable, false, false, null);
```

將佇列設定為耐久後，您可將 `MessageProperties` 設定為 `PERSISTENT_TEXT_PLAIN` (如以下範例所示)，將持續性訊息傳送到您的佇列。

```
import com.rabbitmq.client.MessageProperties;

channel.basicPublish("", "my_queue",
            MessageProperties.PERSISTENT_TEXT_PLAIN,
            message.getBytes());
```

## 讓佇列保持簡短

在叢集部署中，具有大量訊息的佇列可能會導致資源過度使用。當代理程式過度使用時，重新啟動 Amazon MQ for RabbitMQ 代理程式可能會導致效能進一步降低。若已重新啟動，過度使用的代理程式可能會在 `REBOOT_IN_PROGRESS` 狀態中變得沒有回應。

在維護時段期間，Amazon MQ 會一次執行一個節點的所有維護工作，以確保代理程式維持運作狀態。因此，佇列可能需要在每個節點繼續操作時進行同步處理。在同步期間，需要複寫到鏡像的訊息會從對

應的 Amazon Elastic Block Store (Amazon EBS) 磁碟區載入記憶體中，以便分批處理。分批處理訊息可讓佇列更快速地同步處理。

如果佇列保持簡短且訊息很小，則佇列會成功同步處理並繼續如預期般操作。不過，如果批次中的資料量接近節點的記憶體限制，節點就會引發高記憶體警示，暫停佇列同步。您可以比較中的`RabbitMemUsed`和 `RabbitMqMemLimit` Broker 節點測量結果，以確認記憶體使用狀況 CloudWatch。直到取用或刪除訊息或減少批次中的訊息數目，才能完成同步處理。

如果叢集部署暫停佇列同步處理，建議您取用或刪除訊息，以減少佇列中的訊息數目。一旦佇列深度減少且佇列同步完成，代理程式狀態就會變更為 RUNNING。若要解決暫停的佇列同步，您也可套用政策以降低佇列同步處理批次大小。

> **Warning**
>
> 請勿重新啟動資源高漲的代理程式。
> 如果您在佇列同步處理暫停時重新啟動代理程式，代理程式會重新起始同步處理程序，這會在訊息從儲存體傳輸至節點記憶體時進一步降低代理程式資源，並導致代理程式在 `REBOOT_IN_PROGRESS` 狀態中變得沒有回應。

## 設定認可與確認

當用戶端應用程式將訊息的傳遞和取用確認傳送回代理程式時，這稱為消費者確認。同樣地，將確認傳送給發行者的程序也稱為發行者確認。認可和確認對於在使用 RabbitMQ 代理程式時確保資料安全至關重要。

消費者交付認可通常會設定於用戶端應用程式上。使用 AMQP 0-9-1 時，設定 `basic.consume` 或在使用 `basic.code` 方法提取訊息時，可以啟用認可。

通常，交付認可會在通道中啟用。例如，使用 RabbitMQ Java 用戶端程式庫時，您可使用 `Channel#basicAck` 來設定簡單的`basic.ack` 肯定認可，如以下範例所示。

```
// 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**
>
> 未認可的訊息必須在記憶體中快取。您可設定用戶端應用程式的預先擷取設定，以限制消費者預先擷取的訊息數量。

## 設定預先擷取

您可以使用 RabbitMQ 預先擷取值來最佳化消費者取用訊息的方式。RabbitMQ 可將預先擷取計數應用到取消費者而不是通道，以實作 AMQP 0-9-1 提供的通道預先擷取機制。預先擷取值用於指定在任何給定的時間傳送給消費者的訊息數量。根據預設，RabbitMQ 會為用戶端應用程式設定不受限的緩衝區大小。

為 RabbitMQ 消費者設定預先擷取計數時，需要考量各種因素。首先，請考量消費者的環境和組態。因為消費者需要在處理時讓所有訊息保留在記憶體中，因此高預先擷取值可能會對消費者的效能產生負面影響，在某些情況下，可能會導致消費者可能一起當機。同樣地，RabbitMQ 代理程式本身會讓其傳送的所有訊息保持在記憶體中快取，直到其收到消費者認可為止。如果沒有為消費者設定自動認可，且消費者花相對長的時間來處理訊息，則高預先擷取值可能會導致 RabbitMQ 伺服器快速耗盡記憶體。

考慮到上述考量，我們建議一律設定預先擷取值，以防止 RabbitMQ 代理程式或其消費者因大量未處理或未認可的訊息而耗盡記憶體的情況。如果您需要將代理程式最佳化以處理大量訊息，您可使用一系列預先擷取計數來測試代理程式和消費者，以判斷相較於消費者處理訊息所需的時間，網路負荷在哪個時候變得非常微不足道的值。

> **Note**
>
> - 如果用戶端應用程式已設定為自動認可訊息傳遞給消費者，則設定預先擷取值沒有作用。
> - 所有預先擷取的訊息會從佇列中移除。

以下範例示範如何使用 RabbitMQ Java 用戶端程式庫，為單一消費者設定 10 的預先擷取值。

```
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 用戶端程式庫中，`global` 旗標的預設值會設為 `false`，所以上述範例可以簡單地撰寫為 `channel.basicQos(10)`。

## 設定 Celery

Python Celery 會傳送許多不必要的訊息，使搜尋與處理實用資訊變得更加困難。若要減少雜訊並使處理更加輕鬆，請輸入下列命令：

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

## 從網路故障中自動復原

我們建議一律啟用自動網路復原，以避免在用戶端連線到 RabbitMQ 節點失敗的情況下發生重大停機。從版本 `4.0.0` 開始，RabbitMQ Java 用戶端程式庫預設支援自動網路復原。

如果連線的 I/O 迴圈中擲回未處理的例外狀況、如果偵測到通訊端讀取作業逾時，或者伺服器遺漏活動訊號，就會觸發自動連線復原。

在用戶端與 RabbitMQ 節點之間的初始連線失敗的情況下，將不會觸發自動復原。我們建議您透過重試連線來撰寫應用程式程式碼，以解決初始連線失敗的問題。以下範例示範如何使用 RabbitMQ Java 用戶端程式庫重試初始網路故障。

```
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**
>
> 如果應用程式使用 `Connection.Close` 方法關閉連線，則不會啟用或觸發自動網路復原。

## 為您的 RabbitMQ 代理程式啟用傳統佇列 v2

我們建議您為 3.10 或更新版本的代理程式引擎啟用傳統佇列 v2 (CQv2)，以提升效能，包括：

- 將記憶體用量變化平坦化
- 減少大多數工作負載的記憶體用量
- 提升消費者交付
- 增加工作負載輸送量，讓消費者與生產者保持聯繫

若要使用 CQv2，您必須先啟用`classic_mirrored_queue_version`功能旗標。如需有關功能旗標的詳細資訊，請參閱如何啟用功能旗標。

若要從 CQv1 移轉至 CQv2，您必須在原則金鑰定義設為的情況下，建立新的佇列原則或編輯現有的佇列`queue-version`原則。2如需套用原則的詳細資訊，請參閱政策。如需使用佇列政策啟用 CQv2 的詳細資訊，請參閱 RabbitMQ 文件中的傳統佇列。

對於高負載下的佇列，從 CQv1 遷移至 CQv2 可能會佔用大量記憶體。我們建議您在開始遷移之前遵循其他最佳效能實務。

如果您使用佇列政策，刪除佇列政策會將 CQv2 佇列降級回 CQv1。我們不建議將 CQv2 佇列降級為 CQv1，因為 RabbitMQ 會轉換佇列的磁碟上表示法。對於高深度的佇列，這可能會佔用大量記憶體且耗時。

# Amazon MQ for RabbitMQ 中的配額

本主題列出 Amazon MQ 中的配額。您可以針對特定的 AWS 帳戶變更下列許多配額。若要請求提高限制，請參閱 Amazon Web Services 一般參考 中的 AWS Service Quotas。即使套用上限，更新的限制也不會顯示。如需檢視 Amazon CloudWatch 中目前連線限制的詳細資訊，請參閱使用 Amazon CloudWatch 監控 Amazon MQ 代理程式。

主題



## 中介裝置

下表列出與 Amazon MQ for RabbitMQ 代理程式相關的配額。

| 限制 | 描述 |
| --- | --- |
| 代理程式名稱 | • 在代理程式區域中和您的 AWS 帳戶內，必須是唯一的。<br>• 長度必須介於 1 至 50 個字元之間。<br>• 必須僅包含 ASCII 可列印字元集中指定的字元。<br>• 只能包含英數字元、破折號、句點、底線和波狀符號 (- . _ ~)。 |
| 每區域的代理程式數目 | 50 |
| 每個代理程式的安全群組數 | 5 |
| CloudWatch 中監控的 ActiveMQ 目的地 (佇列與主題) | CloudWatch 只會監控前 1000 個目的地。 |

| 限制 | 描述 |
| --- | --- |
| 在 CloudWatch 中監控的 RabbitMQ 目的地 (佇列) | CloudWatch 只會監控前 500 個目的地 (依消費者數量排序)。 |
| 每個代理程式的標籤 | 50 |

## 資料儲存體

下表列出與 Amazon MQ for RabbitMQ 資料儲存相關的配額。

| 限制 | 描述 |
| --- | --- |
| 每個較小代理程式的儲存容量 | `mq.*.micro` 執行個體類型代理程式為 20 GB。如需 Amazon MQ 執行個體類型的詳細資訊，請參閱 Broker instance types。 |
| 每個較大代理程式的儲存容量 | `mq.*.*large` 執行個體類型代理程式為 200 GB。如需 Amazon MQ 執行個體類型的詳細資訊，請參閱 Broker instance types。 |

## API 調節

每個 AWS 帳戶的以下調節配額是跨所有 Amazon MQ API 而彙總的，以維護服務頻寬。如需 Amazon MQ API 的詳細資訊，請參閱 Amazon MQ REST API 參考。

> **Important**
>
> 這些配額不適用於 Amazon MQ for ActiveMQ 或 Amazon MQ for RabbitMQ 代理程式傳訊 API。例如，Amazon MQ 不會調節訊息的傳送或接收。

| API 高載限制 | API 速率限制 |
| --- | --- |
| 100 | 15 |

# Amazon MQ 的安全性

雲端安全是 AWS 最重視的一環。身為 AWS 的客戶，您將能從資料中心和網路架構中獲益，這些都是專為最重視安全的組織而設計的。

安全是 AWS 與您共同肩負的責任。共同的責任模型將其描述為雲端本身的安全和雲端內部的安全：

- 雲端本身的安全 – AWS 負責保護在 AWS Cloud 中執行 AWS 服務的基礎設施。AWS 也提供您可安全使用的服務。第三方稽核人員會定期測試和驗證我們安全性的有效性，做為 AWS 合規計劃的一部分。若要了解適用於 Amazon MQ 的合規計劃，請參閱合規計劃的 AWS 服務範圍。
- 雲端內部的安全 – 您的責任取決於所使用的 AWS 服務。您也必須對其他因素負責，包括資料的機密性、您公司的要求和適用法律和法規。

本文件有助於您了解如何在使用 Amazon MQ 時套用共同責任模型。下列主題說明如何將 Amazon MQ 設定為符合您的安全與合規目標。您也將了解如何使用其他 AWS 服務，幫助您監控並保護 Amazon MQ 資源。

主題

- Amazon MQ 的資料保護
- Amazon MQ 的 Identity and Access Management
- Amazon MQ 的合規驗證
- Amazon MQ 的恢復能力
- Amazon MQ 的基礎設施安全性
- Amazon MQ 的安全最佳實務

# Amazon MQ 的資料保護

AWS 共同的責任模型適用於 Amazon MQ 中的資料保護。如此模型所述，AWS 負責保護執行所有 AWS 雲端 的全球基礎設施。您必須負責維護在此基礎設施上託管之內容的控制權。您也必須負責您所使用的 AWS 服務 安全性設定和管理工作。如需有關資料隱私權的詳細資訊，請參閱資料隱私權常見問答集。如需有關歐洲資料保護的相關資訊，請參閱 AWS 安全性部落格上的 AWS 共同的責任模型和 GDPR 部落格文章。

基於資料保護目的，建議您使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 保護 AWS 帳戶 憑證，並設定個人使用者。如此一來，每個使用者都只會獲得授與完成其任務所必須的許可。我們也建議您採用下列方式保護資料：

- 每個帳戶均要使用多重要素驗證 (MFA)。
- 使用 SSL/TLS 與 AWS 資源通訊。我們需要 TLS 1.2 並建議使用 TLS 1.3。
- 使用 AWS CloudTrail 設定 API 和使用者活動日誌記錄。
- 使用 AWS 加密解決方案，以及 AWS 服務 內的所有預設安全控制項。
- 使用進階的受管安全服務(例如 Amazon Macie)，協助探索和保護儲存在 Amazon S3 的敏感資料。
- 如果您在透過命令列介面或 API 存取 AWS 時，需要 FIPS 140-2 驗證的加密模組，請使用 FIPS 端點。如需有關 FIPS 和 FIPS 端點的詳細資訊，請參閱聯邦資訊處理標準 (FIPS) 140-2 概觀。

我們強烈建議您絕對不要將客戶的電子郵件地址等機密或敏感資訊，放在標籤或自由格式的文字欄位中，例如名稱欄位。這包括當您使用 Amazon MQ 或其他使用主控台、API、AWS CLI 或 AWS SDK 的 AWS 服務 時。您在標籤或自由格式文字欄位中輸入的任何資料都可能用於計費或診斷日誌。如果您提供外部伺服器的 URL，我們強烈建議請勿在驗證您對該伺服器請求的 URL 中包含憑證資訊。

對於 Amazon MQ for ActiveMQ 和 Amazon MQ for RabbitMQ 代理程式，在透過代理程式 Web 主控台或 Amazon MQ API 建立資源時，請勿使用任何個人身分識別資訊 (PII) 或其他機密或敏感資訊作為代理程式名稱或使用者名稱。包括 CloudWatch Logs 在內的其他 AWS 服務可存取代理程式名稱和使用者名稱。代理程式使用者名稱不適用於私有或敏感資料。

## 加密

存放在 Amazon MQ 中的使用者資料會靜態加密。Amazon MQ 靜態加密使用存放在 AWS Key Management Service (KMS) 的加密金鑰將資料加密，以提供加強的安全性。此服務協助降低了保護敏感資料所涉及的操作負擔和複雜性。您可以透過靜態加密，建立符合加密合規和法規要求，而且對安全性要求甚高的應用程式。

所有 Amazon MQ 代理程式之間的連線都使用 Transport Layer Security (TLS) 來提供傳輸中加密。

Amazon MQ 使用其安全管理和儲存的加密金鑰來加密靜態和傳輸中的訊息。如需詳細資訊，請參閱《AWS Encryption SDK 開發人員指南》。

## 靜態加密

Amazon MQ 與 AWS Key Management Service (KMS) 整合，以提供透明的伺服器端加密。Amazon MQ 一律會在靜態時加密您的資料。

當您建立 Amazon MQ for ActiveMQ 代理程式或 Amazon MQ for RabbitMQ 代理程式時，您可以指定您希望 Amazon MQ 用來加密靜態資料的 AWS KMS key。如不指定 KMS 金鑰，Amazon MQ 就會為您建立 AWS 擁有的 KMS 金鑰，並代表您使用它。Amazon MQ 目前支援對稱 KMS 金鑰。如需 KMS 金鑰的詳細資訊，請參閱 AWS KMS keys。

建立代理程式時，您可以選取以下其中一項，以設定哪些 Amazon MQ 用於您的加密金鑰。

- Amazon MQ owned KMS key (default) (Amazon MQ 擁有的 KMS 金鑰 (預設)) — 此金鑰由 Amazon MQ 擁有及管理，不在您的帳戶中。
- AWS 受管 KMS 金鑰 — AWS 受管 KMS 金鑰 (`aws/mq`) 是您帳戶中的 KMS 金鑰，由 Amazon MQ 代表您建立、管理和使用。
- 選取現有客戶管理的 KMS 金鑰 — 客戶管理的 KMS 金鑰由您在 AWS Key Management Service (KMS) 中建立和管理。

> **Important**
>
> - 撤銷授予是無法復原的。如果您需要撤銷存取權限，我們建議您改為刪除代理程式。
> - 針對使用 Amazon Elastic File System (EFS) 存放訊息資料的 Amazon MQ for ActiveMQ 代理程式，如果您撤銷授予允許 Amazon EFS 在您的帳戶中使用 KMS 金鑰的許可，此動作不會立即執行。
> - 針對使用 EBS 存放訊息資料的 Amazon MQ for RabbitMQ 和 Amazon MQ to ActiveMQ 代理程式，如果您停用、排程刪除或撤銷授予允許 Amazon EBS 在您的帳戶中使用 KMS 金鑰的許可，Amazon MQ 將無法維護代理程式，且其可能變更為降級狀態。
> - 如果您已停用金鑰或已排定該金鑰的刪除時程，您可以重新啟用金鑰或取消金鑰刪除作業，保持代理程式的維護狀態。
> - 停用金鑰或撤銷授予不會立即執行。

使用 RabbitMQ 的 KMS 金鑰建立單一執行個體代理程式時，您會看到兩個 `CreateGrant` 事件已登入 AWS CloudTrail。第一個事件是 Amazon MQ 建立 KMS 金鑰的授權。第二個事件是 EBS 建立授權以供 EBS 使用。

CreateGrant AWS CloudTrail 日誌項目：單一執行個體代理程式

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

您將看到一個 EBS 授權建立的事件。

```
                                    {
    "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 RabbitMQ 的 KMS 金鑰建立叢集部署時，您會看到五個已登入的 CreateGrant 事件 AWS CloudTrail。前兩個事件是 Amazon MQ 的授權建立。接下來的三個事件是 EBS 建立的授權，供 EBS 使用。

CreateGrant AWS CloudTrail 日誌項目：叢集部署

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

您將看到三個 EBS 授權建立的事件。

```
                                    {
    "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"
}
```

如需 KMS 金鑰的詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的 AWS KMS keys。

## 傳輸中加密

Amazon MQ for ActiveMQ：Amazon MQ for ActiveMQ 需要強大的 Transport Layer Security (TLS) 並加密在您的 Amazon MQ 部署的代理程式間傳輸中的資料。Amazon MQ 代理程式之間的所有資料都是使用強大的 Transport Layer Security (TLS) 加密。這適用於所有可用的通訊協定。

Amazon MQ for RabbitMQ：Amazon MQ for RabbitMQ 需要對所有用戶端連接進行強大的 Transport Layer Security (TLS) 加密。RabbitMQ 叢集複寫流量只會傳輸代理程式的 VPC，而 AWS 資料中心之

間的所有網路流量都會在實體層透明地加密。Amazon MQ for RabbitMQ 叢集化代理程式目前不支援叢集複寫的節點間加密。若要深入了解傳輸中的資料，請參閱加密靜態和傳輸中的資料。

## Amazon MQ for ActiveMQ 通訊協定

您可以使用已啟用 TLS 的下列通訊協定來存取 ActiveMQ 代理程式：

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

### 針對 ActiveMQ 支援的 TLS 密碼套件

Amazon MQ 上的 ActiveMQ 支援下列密碼套件：

- 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 通訊協定

您可以使用已啟用 TLS 的下列通訊協定來存取 RabbitMQ 代理程式：

- AMQP (0-9-1)

針對 RabbitMQ 支援的 TLS 密碼套件

Amazon MQ 上的 RabbitMQ 支援下列密碼套件：

- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

## Amazon MQ 的 Identity and Access Management

AWS Identity and Access Management (IAM) 是一種 AWS 服務，讓管理員能夠安全地控制對 AWS 資源的存取權限。IAM 管理員可以控制驗證 (已登入) 和授權 (具有許可) 來使用 Amazon MQ 資源。IAM 是一種您可以免費使用的 AWS 服務。

主題

## 對象

AWS Identity and Access Management (IAM) 的使用方式會不同，取決於您在 Amazon MQ 中所執行的工作。

服務使用者 – 如果您使用 Amazon MQ 執行任務，您的管理員會為您提供您需要的登入資料和許可。隨著您為了執行作業而使用的 Amazon MQ 功能數量變多，您可能會需要額外的許可。了解存取許可的管理方式可協助您向管理員請求正確的許可。如果您無法存取 Amazon MQ 中的功能，請參閱 Amazon MQ 身分識別和存取疑難排解。

服務管理員 – 若您在公司負責管理 Amazon MQ 資源，您應該具備 Amazon MQ 的完整存取權限。您的任務是判斷服務使用者應存取的 Amazon MQ 功能和資源。您接著必須將請求提交給您的 IAM 管理員，來變更您服務使用者的許可。檢閱此頁面上的資訊，了解 IAM 的基本概念。若要進一步了解貴公司可搭配 Amazon MQ 使用 IAM 的方式，請參閱 Amazon MQ 如何搭配 IAM 運作。

IAM 管理員 – 如果您是 IAM 管理員，建議您掌握如何撰寫政策以管理 Amazon MQ 存取權的詳細資訊。若要檢視您可以在 IAM 中使用的基於 Amazon MQ 身分的政策範例，請參閱Amazon MQ 身分型政策範例。

## 使用身分來驗證

身分驗證是使用身分憑證登入 AWS 的方式。您必須以 AWS 帳戶根使用者、IAM 使用者身分，或擔任 IAM 角色進行 驗證 (登入至 AWS)。

您可以使用透過身分來源提供的憑證，以聯合身分登入 AWS。AWS IAM Identity Center(IAM Identity Center) 使用者、貴公司的單一登入身分驗證和您的 Google 或 Facebook 憑證都是聯合身分的範例。當您以聯合身分登入時，您的管理員先前已設定使用 IAM 角色的聯合身分。當您 AWS 藉由使用聯合進行存取時，您會間接擔任角色。

根據您的使用者類型，您可以登入 AWS Management Console 或 AWS 存取入口網站。如需有關登入至 AWS 的詳細資訊，請參閱 AWS 登入 使用者指南中的如何登入您的 AWS 帳戶。

如果您是以程式設計的方式存取 AWS，AWS 提供了軟體開發套件 (SDK) 和命令行介面 (CLI)，以便使用您的憑證透過密碼編譯方式簽署您的請求。如果您不使用 AWS 工具，您必須自行簽署請求。如需使用建議的方法自行簽署請求的詳細資訊，請參閱 IAM 使用者指南中的簽署 AWS API 請求。

無論您使用何種身分驗證方法，您可能都需要提供額外的安全性資訊。例如，AWS 建議您使用多重要素驗證 (MFA) 來提高帳戶的安全。如需更多資訊，請參閱 AWS IAM Identity Center 使用者指南中的多重要素驗證和 IAM 使用者指南中的在 AWS 中使用多重要素驗證 (MFA)。

### AWS 帳戶 根使用者

如果是建立 AWS 帳戶，您會先有一個登入身分，可以完整存取帳戶中所有 AWS 服務 與資源。此身分稱為 AWS 帳戶 根使用者，使用建立帳戶時所使用的電子郵件地址和密碼即可登入並存取。強烈建議您不要以根使用者處理日常作業。保護您的根使用者憑證，並將其用來執行只能由根使用者執行的任務。如需這些任務的完整清單，了解需以根使用者登入的任務，請參閱 IAM 使用者指南中的需要根使用者憑證的任務。

### 使用者和群組

IAM 使用者是您 AWS 帳戶 中的一種身分，具備單一人員或應用程式的特定許可。建議您盡可能依賴暫時性憑證，而不是擁有建立長期憑證 (例如密碼和存取金鑰) 的 IAM 使用者。但是如果特定使用案例需要擁有長期憑證的 IAM 使用者，建議您輪換存取金鑰。如需詳細資訊，請參閱 IAM 使用者指南中的為需要長期憑證的使用案例定期輪換存取金鑰。

IAM 群組是一種指定 IAM 使用者集合的身分。您無法以群組身分登入。您可以使用群組來一次為多名使用者指定許可。群組可讓管理大量使用者許可的過程變得更為容易。例如，您可以擁有一個名為 IAMAdmins 的群組，並給予該群組管理 IAM 資源的許可。

使用者與角色不同。使用者只會與單一人員或應用程式建立關聯，但角色的目的是在由任何需要它的人員取得。使用者擁有永久的長期憑證，但角色僅提供臨時憑證。如需進一步了解，請參閱IAM 使用者指南中的建立 IAM 使用者 (而非角色) 的時機。

### IAM 角色

IAM 角色是您 AWS 帳戶 中的一種身分，具備特定許可。它類似 IAM 使用者，但不與特定的人員相關聯。您可以在 AWS Management Console 中透過切換角色來暫時取得 IAM 角色。您可以透過呼叫 AWS CLI 或 AWS API 操作，或是使用自訂 URL 來取得角色。如需使用角色的方法詳細資訊，請參閱 IAM 使用者指南中的使用 IAM 角色。

使用臨時性憑證的 IAM 角色在下列情況中非常有用：

- 聯合身分使用者存取 — 如需向聯合身分指派許可，請建立角色，並為角色定義許可。當聯合身分進行身分驗證時，該身分會與角色建立關聯，並獲授予由角色定義的許可。如需有關聯合角色的詳細資訊，請參閱 IAM 使用者指南中的為第三方身分供應商建立角色。如果您使用 IAM Identity Center，則需要設定許可集。為控制身分驗證後可以存取的內容，IAM Identity Center 將許可集與 IAM 中的角色相關聯。如需有關許可集的資訊，請參閱 AWS IAM Identity Center 使用者指南中的許可集。
- 暫時 IAM 使用者許可：使用者可以擔任 IAM 角色或角色來暫時針對特定任務採用不同的許可。
- 跨帳戶存取權：您可以使用 IAM 角色，允許不同帳戶中的某人 (信任的主體) 存取您帳戶的資源。角色是授予跨帳戶存取權的主要方式。但是，針對某些 AWS 服務，您可以將政策直接連接到資源 (而

非使用角色作為代理)。如需了解使用角色和資源型政策進行跨帳戶存取之間的差異，請參閱IAM 使用者指南中的 IAM 角色與資源類型政策的差異。

- 跨服務存取權：有些 AWS 服務 會使用其他 AWS 服務 中的功能。例如，當您在服務中進行呼叫時，該服務通常會在 Amazon EC2 中執行應用程式或將物件存放在 Amazon Simple Storage Service (Amazon S3) 中。服務可能會使用呼叫主體的許可、使用服務角色或使用服務連結角色來執行此作業。
  - 轉發存取工作階段 (FAS)：當您使用 IAM 使用者或角色在 AWS 中執行動作時，系統會將您視為主體。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS 使用主體的許可呼叫 AWS 服務，搭配請求 AWS 服務以向下游服務發出請求。只有在服務收到需要與其他 AWS 服務或資源互動才能完成的請求之後，才會提出 FAS 請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出 FAS 請求時的政策詳細資訊，請參閱《轉發存取工作階段》https://docs.aws.amazon.com/IAM/latest/UserGuide/access_forward_access_sessions.html。
  - 服務角色 – 服務角色是服務擔任的 IAM 角色，可代表您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需詳細資訊，請參閱 IAM 使用者指南中的建立角色以委派許可給 AWS 服務 服務。
  - 服務連結角色：服務連結角色是一種連結到 AWS 服務 的服務角色類型。服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的 AWS 帳戶 中，並由該服務所擁有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。
- 在 Amazon EC2 上執行的應用程式：針對在 EC2 執行個體上執行並提出 AWS CLI 和 AWS API 請求的應用程式，您可以使用 IAM 角色來管理臨時性憑證。這是在 EC2 執行個體內存放存取金鑰的較好方式。如需指派 AWS 角色給 EC2 執行個體並提供其所有應用程式使用，您可以建立連接到執行個體的執行個體設定檔。執行個體設定檔包含該角色，並且可讓 EC2 執行個體上執行的程式取得臨時性憑證。如需詳細資訊，請參閱 IAM 使用者指南中的利用 IAM 角色來授予許可給 Amazon EC2 執行個體上執行的應用程式。

如需了解是否要使用 IAM 角色或 IAM 使用者，請參閱 IAM 使用者指南中的建立 IAM 角色 (而非使用者) 的時機。

## 使用政策管理存取權

您可以透過建立政策並將其連接到 AWS 身分或資源，在 AWS 中控制存取。政策是 AWS 中的一個物件，當其和身分或資源建立關聯時，便可定義其許可。AWS 會在主體 (使用者、根使用者或角色工作階段) 發出請求時評估這些政策。政策中的許可決定是否允許或拒絕請求。大部分政策以 JSON 文件形式存放在 AWS 中。如需 JSON 政策文件結構和內容的詳細資訊，請參閱 IAM 使用者指南中的 JSON 政策概觀。

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼条件下可以對什麼資源執行哪些動作。

根據預設，使用者和角色沒有許可。若要授予使用者對其所需資源執行動作的許可，IAM 管理員可以建立 IAM 政策。然後，管理員可以將 IAM 政策新增至角色，使用者便能擔任這些角色。

IAM 政策定義該動作的許可，無論您使用何種方法來執行操作。例如，假設您有一個允許 `iam:GetRole` 動作的政策。具備該政策的使用者便可以從 AWS Management Console、AWS CLI 或 AWS API 取得角色資訊。

### 身分型政策

身分型政策是可以附加到身分 (例如 IAM 使用者、使用者群組或角色) 的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。若要了解如何建立身分類型政策，請參閱《IAM 使用者指南》中的建立 IAM 政策。

身分型政策可進一步分類成內嵌政策或受管政策。內嵌政策會直接內嵌到單一使用者、群組或角色。受管政策則是獨立的政策，您可以將這些政策連接到 AWS 帳戶 中的多個使用者、群組和角色。受管政策包含 AWS 管理政策和客戶管理政策。如需了解如何在受管政策及內嵌政策間選擇，請參閱 IAM 使用者指南中的在受管政策和內嵌政策間選擇。

### 資源型政策

資源型政策是連接到資源的 JSON 政策文件。資源型政策的最常見範例是 Amazon Simple Storage Service (Amazon S3) 儲存貯體政策和 IAM 角色信任政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中指定主體。主體可以包括帳戶、使用者、角色、聯合身分使用者或 AWS 服務。

資源型政策是位於該服務中的內嵌政策。您無法在資源型政策中使用來自 IAM 的 AWS 受管政策。

### 存取控制清單 (ACL)

存取控制清單 (ACL) 可控制哪些主體 (帳戶成員、使用者或角色) 擁有存取某資源的許可。ACL 類似於資源型政策，但它們不使用 JSON 政策文件格式。

Amazon Simple Storage Service (Amazon S3)、AWS WAF 和 Amazon VPC 是支援 ACL 的服務範例。如需進一步了解 ACL，請參閱 Amazon Simple Storage Service 開發人員指南中的存取控制清單 (ACL) 概觀。

### 其他政策類型

AWS 支援其他較少見的政策類型。這些政策類型可設定較常見政策類型授予您的最大許可。

- 許可界限：許可界限是一種進階功能，可供您設定身分型政策能授予 IAM 實體 (IAM 使用者或角色) 的最大許可。您可以為實體設定許可界限。所產生的許可會是實體的身分型政策和其許可界限的交集。會在 `Principal` 欄位中指定使用者或角色的資源型政策則不會受到許可界限限制。所有這類政策中的明確拒絕都會覆寫該允許。如需許可界限的詳細資訊，請參閱 IAM 使用者指南中的 IAM 實體許可界限。
- 服務控制政策 (SCP)：SCP 是 JSON 政策，可指定 AWS Organizations 中組織或組織單位 (OU) 的最大許可。AWS Organizations 服務可用來分組和集中管理您企業所擁有的多個 AWS 帳戶。若您啟用組織中的所有功能，您可以將服務控制政策 (SCP) 套用到任何或所有帳戶。SCP 會限制成員帳戶中實體的許可，包括每個 AWS 帳戶根使用者。如需 Organizations 和 SCP 的詳細資訊，請參閱 AWS Organizations 使用者指南中的 SCP 運作方式。
- 工作階段政策：工作階段政策是一種進階政策，您可以在透過編寫程式的方式建立角色或聯合身分使用者的暫時工作階段時，作為參數傳遞。所產生工作階段的許可會是使用者或角色的身分型政策和工作階段政策的交集。許可也可以來自資源型政策。所有這類政策中的明確拒絕都會覆寫該允許。如需詳細資訊，請參閱 IAM 使用者指南中的工作階段政策。

### 多種政策類型

將多種政策類型套用到請求時，其結果形成的許可會更為複雜、更加難以理解。若要了解 AWS 在涉及多種政策類型時如何判斷是否允許一項請求，請參閱《IAM 使用者指南》中的政策評估邏輯。

## Amazon MQ 如何搭配 IAM 運作

在您使用 IAM 管理對 Amazon MQ 的存取權之前，您應該瞭解哪些 IAM 功能可以與 Amazon MQ 搭配使用。若要取得 Amazon MQ 和其他 AWS 服務如何使用 IAM 的詳細資訊，請參閱 IAM 使用者指南中的使用 IAM 的 AWS 服務。

Amazon MQ 使用 IAM 來建立、更新和刪除操作，但對代理程式使用原生 ActiveMQ 身分驗證。如需更多詳細資訊，請參閱 整合 ActiveMQ 代理程式與 LDAP。

主題

- Amazon MQ 以身分為基礎的政策
- Amazon MQ 以資源為基礎的政策
- 以 Amazon MQ 標籤為基礎的授權

- Amazon MQ IAM 角色

## Amazon MQ 以身分為基礎的政策

使用 IAM 身分型政策，您可以指定允許或拒絕的動作和資源，以及在何種條件下允許或拒絕動作。Amazon MQ 支援特定動作、資源和條件金鑰。若要了解您在 JSON 政策中使用的所有元素，請參閱《IAM 使用者指南》中的 JSON 政策元素參考。

動作

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

JSON 政策的 `Action` 元素描述您可以用來允許或拒絕政策中存取的動作。政策動作的名稱通常會和相關聯的 AWS API 操作相同。有一些例外狀況，例如沒有相符的 API 作業的僅限許可動作。也有一些作業需要政策中的多個動作。這些額外的動作稱為相依動作。

政策會使用動作來授予執行相關聯操作的許可。

Amazon MQ 中的政策動作會在動作之前使用下列前綴：`mq:`。例如，若要授予某人使用 Amazon MQ `CreateBroker` API 操作來執行 Amazon MQ 執行個體的許可，請在其政策中加入 `mq:CreateBroker` 動作。政策陳述式必須包含 `Action` 或 `NotAction` 元素。Amazon MQ 會定義自己的一組動作，描述您可以使用此服務執行的任務。

若要在單一陳述式中指定多個動作，請用逗號分隔，如下所示：

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

您也可以使用萬用字元 (*) 來指定多個動作。例如，若要指定開頭是 `Describe` 文字的所有動作，請包含以下動作：

```
"Action": "mq:Describe*"
```

若要查看 Amazon MQ 動作清單，請參閱《IAM 使用者指南》中 Amazon MQ 定義的動作。

資源

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

Resource JSON 政策元素可指定要套用動作的物件。陳述式必須包含 Resource 或 NotResource 元素。最佳實務是使用其 Amazon Resource Name (ARN) 來指定資源。您可以針對支援特定資源類型的動作 (稱為資源層級許可) 來這麼做。

對於不支援資源層級許可的動作 (例如列出作業)，請使用萬用字元 (*) 來表示陳述式適用於所有資源。

```
"Resource": "*"
```

在 Amazon MQ 中，住要 AWS 資源是 Amazon MQ 訊息代理程式及其組態。Amazon MQ 代理程式與組態都有獨一無二的 Amazon 資源名稱 (ARN) 與其相關聯，如下表所示。

| 資源類型 | ARN | 條件索引鍵 |
| --- | --- | --- |
| 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} |

如需 ARN 格式的詳細資訊，請參閱 Amazon 資源名稱 (ARN) 和 AWS 服務命名空間。

例如，若要在您的陳述式中指定名為 MyBroker 且 brokerId 為 b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9 的代理程式，請使用以下 ARN：

```
"Resource": "arn:aws:mq:us-
east-1:123456789012:broker:MyBroker:b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9"
```

若要指定屬於特定帳戶的所有代理程式和組態，請使用萬用字元 (*)：

```
"Resource": "arn:aws:mq:us-east-1:123456789012:*"
```

有些 Amazon MQ 動作無法對特定資源執行，例如用來建立資源的動作。在這些情況下，您必須使用萬用字元 (*)。

```
"Resource": "*"
```

API 動作 CreateTags 同時需要代理程式和組態。若要在單一陳述式中指定多項資源，請使用逗號分隔 ARN。

```
"Resource": [
      "resource1",
      "resource2"
```

若要查看 Amazon MQ 資源類型及其 ARN 的清單，請參閱《IAM 使用者指南》中的 Amazon MQ 定義的資源 。若要了解您可以使用哪些動作指定每個資源的 ARN，請參閱 Amazon MQ 定義的動作。

條件金鑰

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

Condition 元素 (或 Condition 區塊) 可讓您指定使陳述式生效的條件。Condition 元素是選用項目。您可以建立使用條件運算子的條件運算式 (例如等於或小於)，來比對政策中的條件和請求中的值。

若您在陳述式中指定多個 Condition 元素，或是在單一 Condition 元素中指定多個索引鍵，AWS 會使用邏輯 AND 操作評估他們。若您為單一條件索引鍵指定多個值，AWS 會使用邏輯 OR 操作評估條件。必須符合所有條件，才會授予陳述式的許可。

您也可以在指定條件時使用預留位置變數。例如，您可以只在使用者使用其 IAM 使用者名稱標記時，將存取資源的許可授予該 IAM 使用者。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM 政策元素：變數和標籤。

AWS 支援全域條件索引鍵和服務特定的條件索引鍵。若要查看 AWS 全域條件索引鍵，請參閱《IAM 使用者指南》中的 AWS 全域條件內容金鑰。

Amazon MQ 未定義任何服務專用條件索引鍵，但支援一些全域條件索引鍵的使用。若要查看 Amazon MQ 條件金鑰的清單，請參閱《IAM 使用者指南》中的 Amazon MQ 條件索引鍵。若要了解您可以搭配哪些動作和資源使用條件金鑰，請參閱 Amazon MQ 定義的動作。

| 條件索引鍵 | 描述 | 類型 |
| --- | --- | --- |
| aws:RequestTag/${TagKey} | 根據要求中傳遞的標籤篩選動作。 | 字串 |

| 條件索引鍵 | 描述 | 類型 |
| --- | --- | --- |
| aws:ResourceTag/${TagKey} | 根據與資源相關聯的標籤篩選動作。 | 字串 |
| aws:TagKeys | 根據要求中傳遞的標籤鍵篩選動作。 | 字串 |

範例

若要檢視 Amazon MQ 身分類型政策的範例，請參閱Amazon MQ 身分型政策範例。

## Amazon MQ 以資源為基礎的政策

目前，Amazon MQ 不支援使用資源型許可或資源型政策，進行 IAM 身分驗證。

## 以 Amazon MQ 標籤為基礎的授權

您可以將標籤連接至 Amazon MQ 資源，或是在請求中將標籤傳遞至 Amazon MQ。若要根據標籤控制存取，請使用 `mq:ResourceTag/key-name`、`aws:RequestTag/key-name` 或 `aws:TagKeys` 條件索引鍵，在政策的條件元素中，提供標籤資訊。

Amazon MQ 支援以標籤為基礎的政策。例如，您可以拒絕包含索引鍵為 `environment`，值為 `production` 之標籤的 Amazon MQ 資源存取。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "mq:DeleteBroker",
                "mq:RebootBroker",
                "mq:DeleteTags"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/environment": "production"
                }
```

```
            }
        }
    ]
}
```

此政策將 Deny 以下功能：刪除或重新啟動包含標籤 environment/production 的 Amazon MQ 代理程式。

如需標記的詳細資訊，請參閱：

- 標記 資源
- 使用 IAM 標籤控制存取

### Amazon MQ IAM 角色

IAM 角色是您 AWS 帳戶中具備特定許可的實體。

搭配使用暫時登入資料與 Amazon MQ

您可以搭配聯合使用臨時憑證、擔任 IAM 角色，或是擔任跨帳戶角色。您取得臨時安全憑證的方式是透過呼叫 AWS STS API 操作 (例如，AssumeRole 或 GetFederationToken)。

Amazon MQ 支援使用臨時登入資料。

服務角色

此功能可讓服務代表您擔任服務角色。此角色可讓服務存取其他服務中的資源，以代表您完成動作。服務角色會出現在您的 IAM 帳戶中，且由該帳戶所擁有。這表示 IAM 管理員可以變更此角色的許可。不過，這樣可能會破壞此服務的功能。

Amazon MQ 支援服務角色。

## Amazon MQ 身分型政策範例

根據預設，IAM 使用者和角色不具備建立或修改 Amazon MQ 資源的許可。他們也無法使用 AWS Management Console、AWS CLI 或 AWS API 執行任務。IAM 管理員必須建立 IAM 政策，授予使用者和角色在指定資源上執行特定 API 操作的所需許可。管理員接著必須將這些政策連接至需要這些許可的 IAM 使用者或群組。

若要了解如何使用這些範例 JSON 政策文件建立 IAM 身分型政策，請參閱《IAM 使用者指南》中的在 JSON 標籤上建立政策。

主題



## 政策最佳實務

身分型政策會判斷您帳戶中的某個人員是否可以建立、存取或刪除 Amazon MQ 資源。這些動作可能會讓您的 AWS 帳戶 產生費用。當您建立或編輯身分型政策時，請遵循下列準則及建議事項：

- 開始使用 AWS 受管政策並朝向最低權限許可的目標邁進 – 若要開始授予許可給使用者和工作負載，請使用 AWS 受管政策，這些政策會授予許可給許多常用案例。它們可在您的 AWS 帳戶 中使用。我們建議您定義特定於使用案例的 AWS 客戶管理政策，以便進一步減少許可。如需詳細資訊，請參閱 IAM 使用者指南中的 AWS 受管政策或任務職能的 AWS 受管政策。
- 套用最低權限許可 – 設定 IAM 政策的許可時，請僅授予執行任務所需的許可。為實現此目的，您可以定義在特定條件下可以對特定資源採取的動作，這也稱為最低權限許可。如需使用 IAM 套用許可的詳細資訊，請參閱 IAM 使用者指南中的 IAM 中的政策和許可。
- 使用 IAM 政策中的條件進一步限制存取權 – 您可以將條件新增至政策，以限制動作和資源的存取。例如，您可以撰寫政策條件，指定必須使用 SSL 傳送所有請求。您也可以使用條件來授予對服務動作的存取權，前提是透過特定 AWS 服務 (例如 AWS CloudFormation) 使用條件。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM JSON 政策元素：條件。
- 使用 IAM Access Analyzer 驗證 IAM 政策，確保許可安全且可正常運作 – IAM Access Analyzer 驗證新政策和現有政策，確保這些政策遵從 IAM 政策語言 (JSON) 和 IAM 最佳實務。IAM Access Analyzer 提供 100 多項政策檢查及切實可行的建議，可協助您編寫安全且實用的政策。如需詳細資訊，請參閱 IAM 使用者指南中的 IAM Access Analyzer 政策驗證。
- 需要多重要素驗證 (MFA) – 如果存在需要 AWS 帳戶中 IAM 使用者或根使用者的情況，請開啟 MFA 提供額外的安全性。若要在呼叫 API 操作時請求 MFA，請將 MFA 條件新增至您的政策。如需詳細資訊，請參閱 IAM 使用者指南中的設定 MFA 保護的 API 存取。

有關 IAM 中最佳實務的詳細資訊，請參閱 IAM 使用者指南中的 IAM 安全最佳實務。

## 使用 Amazon MQ 主控台

若要存取 Amazon MQ 主控台，您必須擁有最基本的一組許可。這些許可必須允許您列出和檢視您 AWS 帳戶中 Amazon MQ 資源的詳細資訊。如果您建立比最基本必要許可更嚴格的身分型政策，則對於具有該政策的實體 (IAM 使用者或角色) 而言，主控台就無法如預期運作。

為確保那些實體仍可使用 Amazon MQ 主控台，請也將以下 AWS 管理的政策連接到實體。如需詳細資訊，請參閱《IAM 使用者指南》中的新增許可到使用者。

```
AmazonMQReadOnlyAccess
```

對於僅呼叫 AWS CLI 或 AWS API 的使用者，您不需要允許其最基本主控台許可。反之，只需允許存取符合您嘗試執行之 API 操作的動作就可以了。

## 允許使用者檢視他們自己的許可

此範例會示範如何建立政策，允許 IAM 使用者檢視連接到他們使用者身分的內嵌及受管政策。此政策包含在主控台上，或是使用 AWS CLI 或 AWS API 透過編寫程式的方式完成此動作的許可。

```
{
    "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 身分驗證和授權

Amazon MQ 會使用標準 AWS 請求簽署，進行 API 身分驗證。如需詳細資訊，請參閱《AWS 一般參考》AWS中的簽署 API 請求。

> **Note**
>
> 目前，Amazon MQ 不支援使用資源型許可或資源型政策，進行 IAM 身分驗證。

若要授權 AWS 使用者可以使用代理程式、組態和使用者，您必須編輯 IAM 政策許可。

主題

- 建立 Amazon MQ 代理程式所需的 IAM 許可
- Amazon MQ REST API 許可參考
- Amazon MQ API 動作的資源層級許可

## 建立 Amazon MQ 代理程式所需的 IAM 許可

若要建立代理程式，您必須使用 `AmazonMQFullAccess` IAM 政策或是將下列 EC2 許可納入 IAM 政策。

以下自訂政策包含兩個陳述式 (一個條件式)，其會授予許可以操作 Amazon MQ 建立 ActiveMQ 代理程式所需的資源。

> **Important**
>
> - 需要 `ec2:CreateNetworkInterface` 動作，才能允許 Amazon MQ 代表您在帳戶中建立彈性網路界面 (ENI)。
> - `ec2:CreateNetworkInterfacePermission` 動作會授權 Amazon MQ，將 ENI 連接到 ActiveMQ 代理程式。
> - `ec2:AuthorizedService` 條件金鑰可確保僅能將 ENI 許可授予 Amazon MQ 服務帳戶。

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

如需詳細資訊，請參閱 步驟 2：建立使用者並取得您的 AWS 登入資料 及 永不修改或刪除 Amazon MQ 彈性網路界面。

## Amazon MQ REST API 許可參考

下表列出 Amazon MQ REST API 和對應的 IAM 許可。

## Amazon MQ REST API 和必要許可

| Amazon MQ REST API | 所需的許可 |
| --- | --- |
| CreateBroker | mq:CreateBroker |
| CreateConfiguration | mq:CreateConfiguration |
| CreateTags | mq:CreateTags |
| CreateUser | mq:CreateUser |
| DeleteBroker | mq:DeleteBroker |
| DeleteUser | mq:DeleteUser |
| DescribeBroker | mq:DescribeBroker |
| DescribeConfiguration | mq:DescribeConfiguration |
| DescribeConfigurationRevision | mq:DescribeConfigurationRevision |
| DescribeUser | mq:DescribeUser |
| ListBrokers | mq:ListBrokers |
| ListConfigurationRevisions | mq:ListConfigurationRevisions |
| ListConfigurations | mq:ListConfigurations |
| ListTags | mq:ListTags |
| ListUsers | mq:ListUsers |
| RebootBroker | mq:RebootBroker |
| UpdateBroker | mq:UpdateBroker |
| UpdateConfiguration | mq:UpdateConfiguration |
| UpdateUser | mq:UpdateUser |

# Amazon MQ API 動作的資源層級許可

資源層級許可一詞是指能夠讓您指定使用者可執行動作的資源。Amazon MQ 支援部分的資源層級許可。針對特定 Amazon MQ 動作，您可以根據應滿足的條件，或允許使用者使用特定的資源，控制使用者何時可以使用那些動作。

下表說明目前支援資源層級許可的 Amazon MQ API 動作，以及每個動作支援的資源、資源 ARN 和條件金鑰。

> **Important**
>
> 若在此資料表中並未列出某個 Amazon MQ API 動作，表示該動作目前不支援資源層級許可。若 Amazon MQ API 動作不支援資源層級許可，您可以授予使用者使用此動作的許可，但您必須針對政策陳述式中的資源元素指定 * 萬用字元。

| API 動作 | 資源類型 (*必填項目) |
| --- | --- |
| CreateConfiguration | 組態* |
| CreateTags | 代理程式 、組態 |
| CreateUser | 中介裝置* |
| DeleteBroker | 中介裝置* |
| DeleteUser | 中介裝置* |
| DescribeBroker | 中介裝置* |
| DescribeConfiguration | 組態* |
| DescribeConfigurationRevision | 組態* |
| DescribeUser | 中介裝置* |
| ListConfigurationRevisions | 組態* |

| API 動作 | 資源類型 (*必填項目) |
| --- | --- |
| ListConfigurationRevisions | 組態* |
| ListTags | 代理程式、組態 |
| ListUsers | 中介裝置* |
| RebootBroker | 中介裝置* |
| UpdateBroker | 中介裝置* |
| UpdateConfiguration | 組態* |
| UpdateUser | 中介裝置* |

## Amazon MQ 的 AWS 受管政策

AWS 受管政策是由 AWS 建立和管理的獨立政策。AWS 受管政策的設計在於為許多常見使用案例提供許可，如此您就可以開始將許可指派給使用者、群組和角色。

請記住，AWS 受管政策可能不會授與您特定使用案例的最低權限許可，因為它們可供所有 AWS 客戶使用。我們建議您定義使用案例專屬的客戶管理政策，以便進一步減少許可。

您無法更改 AWS 受管政策中定義的許可。如果 AWS 更新 AWS 受管政策中定義的許可，更新會影響政策連接的所有主體實體 (使用者、群組和角色)。在推出新的 AWS 服務 或有新的 API 操作可供現有服務使用時，AWS 很可能會更新 AWS 受管政策。

如需詳細資訊，請參閱《IAM 使用者指南》https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_managed-vs-inline.html#aws-managed-policies中的 AWS 受管政策。

### AWS 受管政策：AmazonMQServiceRolePolicy

您不得將 AmazonMQServiceRolePolicy 附加至 IAM 實體。此政策會連接到服務連結角色，而此角色可讓 Amazon MQ 代表您執行動作。如需此許可政策及其允許 Amazon MQ 執行之動作的詳細資訊，請參閱 the section called “Amazon MQ 的服務連結角色許可”。

### Amazon MQ 的 AWS 受管政策更新

檢視自此服務開始追蹤 Amazon MQ AWS 受管政策更新以來的變更詳細資訊。如需有關此頁面變更的自動提醒，請訂閱 Amazon MQ Document history (文件歷程記錄) 頁面上的 RSS 摘要。

| 變更 | 描述 | 日期 |
| --- | --- | --- |
| Amazon MQ 開始追蹤變更 | Amazon MQ 開始追蹤其 AWS 受管政策的變更 | 2021 年 5 月 5 日 |

## 使用 Amazon MQ 的服務連結角色

Amazon MQ 使用 AWS Identity and Access Management (IAM) 服務連結角色。服務連結角色是直接連結至 Amazon MQ 的一種特殊 IAM 角色類型。服務連結角色由 Amazon MQ 預先定義，內含該服務代您呼叫其他 AWS 服務所需的所有許可。

服務連結角色可讓設定 Amazon MQ 更為簡單，因為您不必手動新增必要的許可。Amazon MQ 定義其服務連結角色的許可，除非另有定義，否則僅有 Amazon MQ 可以擔任其角色。定義的許可包括信任政策和許可政策，並且該許可政策不能連接到任何其他 IAM 實體。

您必須先刪除服務連結角色的相關資源，才能將其刪除。如此可保護您的 Amazon MQ 資源，避免您不小心移除資源的存取許可。

如需關於支援服務連結角色的其他服務的資訊，請參閱可搭配 IAM 運作的 AWS 服務，並尋找 Service-Linked Role (服務連結角色) 欄顯示為 Yes (是) 的服務。選擇具有連結的 Yes (是)，以檢視該服務的服務連結角色文件。

### Amazon MQ 的服務連結角色許可

Amazon MQ 使用名為 AWSServiceRoleForAmazonMQ 的服務連結角色 — Amazon MQ 會使用此服務連結角色代表您呼叫 AWS 服務。

AWSServiceRoleForAmazonMQ 服務連結角色信任下列服務可擔任該角色：

- `mq.amazonaws.com`

Amazon MQ 使用許可政策 `AmazonMQServiceRolePolicy`，其附加至 AWSServiceRoleForAmazonMQ 服務連結角色，以在指定的資源上完成下列動作：

- 動作：vpc 資源上的 ec2:CreateVpcEndpoint。
- 動作：subnet 資源上的 ec2:CreateVpcEndpoint。
- 動作：security-group 資源上的 ec2:CreateVpcEndpoint。
- 動作：vpc-endpoint 資源上的 ec2:CreateVpcEndpoint。
- 動作：vpc 資源上的 ec2:DescribeVpcEndpoints。
- 動作：subnet 資源上的 ec2:DescribeVpcEndpoints。
- 動作：vpc-endpoint 資源上的 ec2:CreateTags。
- 動作：log-group 資源上的 logs:PutLogEvents。
- 動作：log-group 資源上的 logs:DescribeLogStreams。
- 動作：log-group 資源上的 logs:DescribeLogGroups。
- 動作：log-group 資源上的 CreateLogStream。
- 動作：log-group 資源上的 CreateLogGroup。

當您建立 Amazon MQ for RabbitMQ 代理程式時，AmazonMQServiceRolePolicy 許可政策允許 Amazon MQ 代表您執行下列任務。

- 使用您提供的 Amazon VPC、子網路和安全群組，為代理程式建立 Amazon VPC 端點。您可以使用為代理程式建立的端點，透過 RabbitMQ 管理主控台、管理 API 或以程式設計方式連線到代理程式。
- 建立日誌群組，並將代理程式日誌發佈至 Amazon CloudWatch Logs。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "ec2:DescribeVpcEndpoints"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateVpcEndpoint"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:vpc/*",
                "arn:aws:ec2:*:*:subnet/*",
                "arn:aws:ec2:*:*:security-group/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateVpcEndpoint"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:vpc-endpoint/*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/AMQManaged": "true"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": "arn:aws:ec2:*:*:vpc-endpoint/*",
            "Condition": {
                "StringEquals": {
                    "ec2:CreateAction": "CreateVpcEndpoint"
                }
            }
        },
        {
            "Effect": "Allow",
```

```
            "Action": [
                "ec2:DeleteVpcEndpoints"
            ],
            "Resource": "arn:aws:ec2:*:*:vpc-endpoint/*",
            "Condition": {
                "StringEquals": {
                    "ec2:ResourceTag/AMQManaged": "true"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents",
                "logs:DescribeLogStreams",
                "logs:DescribeLogGroups",
                "logs:CreateLogStream",
                "logs:CreateLogGroup"
            ],
            "Resource": [
                "arn:aws:logs:*:*:log-group:/aws/amazonmq/*"
            ]
        }
    ]
}
```

您必須設定許可，IAM 實體 (如使用者、群組或角色) 才可建立、編輯或刪除服務連結角色。如需詳細資訊，請參閱 IAM 使用者指南中的服務連結角色許可。

## 建立 Amazon MQ 的服務連結角色

您不需要手動建立一個服務連結角色。當您第一次建立代理程式時，Amazon MQ 會建立服務連結角色以代表您呼叫 AWS 服務。您建立的所有後續代理程式都會使用相同的角色，而且不會建立新角色。

> **Important**
>
> 此服務連結角色可以顯示在您的帳戶，如果您於其他服務中完成一項動作時，可以使用支援此角色的功能。若要進一步了解，請參閱我的 IAM 帳戶中出現的新角色。

若您刪除此服務連結角色，之後需要再次建立，您可以在帳戶中使用相同程序重新建立角色。

您也可以使用 IAM 主控台，透過 Amazon MQ 使用案例建立服務連結角色。在 AWS CLI CLI 或 AWS API 中，建立一個服務名稱為 mq.amazonaws.com 的服務連結角色。如需詳細資訊，請參閱《IAM 使用者指南》中的建立服務連結角色。如果您刪除此服務連結角色，您可以使用此相同的程序以再次建立該角色。

## 編輯 Amazon MQ 的服務連結角色

Amazon MQ 不允許您編輯 AWSServiceRoleForAmazonMQ 服務連結角色。然而，您可使用 IAM 來編輯角色描述。如需詳細資訊，請參閱《IAM 使用者指南》中的編輯服務連結角色。

## 刪除 Amazon MQ 的服務連結角色

若您不再使用需要服務連結角色的功能或服務，我們建議您刪除該角色。如此一來，您就沒有未主動監控或維護的未使用實體。然而，在手動刪除服務連結角色之前，您必須先清除資源。

> **Note**
>
> 若 Amazon MQ 服務在您試圖刪除資源時正在使用該角色，刪除可能會失敗。若此情況發生，請等待數分鐘後並再次嘗試操作。

刪除 AWSServiceRoleForAmazonMQ 所使用的 Amazon MQ 資源

- 使用 AWS Management Console、Amazon MQ CLI 或 Amazon MQ API 刪除您的 Amazon MQ 代理程式。如需刪除使用者的詳細資訊，請參閱 ???。

使用 IAM 手動刪除服務連結角色

使用 IAM 主控台、AWS CLI 或 AWS API 來刪除 AWSServiceRoleForAmazonMQ 服務連結角色。如需詳細資訊，請參閱 IAM 使用者指南中的刪除服務連結角色。

## Amazon MQ 服務連結角色的支援區域

Amazon MQ 在所有提供服務的區域中支援使用服務連結的角色。如需詳細資訊，請參閱 AWS 區域與端點。

| 區域名稱 | 區域身分 | Amazon MQ 支援 |
| --- | --- | --- |
| 美國東部 (維吉尼亞北部) | us-east-1 | 是 |

| 區域名稱 | 區域身分 | Amazon MQ 支援 |
| --- | --- | --- |
| 美國東部 (俄亥俄) | us-east-2 | 是 |
| 美國西部 (加利佛尼亞北部) | us-west-1 | 是 |
| 美國西部 (奧勒岡) | us-west-2 | 是 |
| 亞太區域 (孟買) | ap-south-1 | 是 |
| 亞太區域 (大阪) | ap-northeast-3 | 是 |
| 亞太區域 (首爾) | ap-northeast-2 | 是 |
| 亞太區域 (新加坡) | ap-southeast-1 | 是 |
| 亞太區域 (雪梨) | ap-southeast-2 | 是 |
| 亞太區域 (東京) | ap-northeast-1 | 是 |
| 加拿大 (中部) | ca-central-1 | 是 |
| 歐洲 (法蘭克福) | eu-central-1 | 是 |
| 歐洲 (愛爾蘭) | eu-west-1 | 是 |
| 歐洲 (倫敦) | eu-west-2 | 是 |
| 歐洲 (巴黎) | eu-west-3 | 是 |
| 南美洲 (聖保羅) | sa-east-1 | 是 |
| AWS GovCloud (US) | us-gov-west-1 | 否 |

# Amazon MQ 身分識別和存取疑難排解

請使用以下資訊來協助您診斷和修正使用 Amazon MQ 和 IAM 時可能遇到的常見問題。

**主題**

- 我未獲授權，不得在 Amazon MQ 中執行動作

- 我未獲得執行 iam: PassRole 的授權
- 我想允許 AWS 帳戶外的人員存取我的 Amazon MQ 資源

## 我未獲授權，不得在 Amazon MQ 中執行動作

若 AWS Management Console 告知您並未獲得執行動作的授權，您必須聯絡您的管理員以取得協助。您的管理員是為您提供登入憑證的人員。

以下範例錯誤會在 `mateojackson` 使用者嘗試使用主控台檢視 *`Widget`* 的詳細資訊，但卻沒有 `mq:`*`GetWidget`* 許可時發生。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 mq:GetWidget on resource: my-example-widget
```

在此情況下，Mateo 會請求管理員更新他的政策，允許他使用 *`my-example-widget`* 動作存取 `mq:`*`GetWidget`* 資源。

## 我未獲得執行 iam: PassRole 的授權

如果您收到錯誤，告知您無權執行 `iam:PassRole` 動作，您的政策必須更新，允許您將角色傳遞給 Amazon MQ。

有些 AWS 服務 允許您傳遞現有的角色至該服務，而無須建立新的服務角色或服務連結角色。若要執行此作業，您必須擁有將角色傳遞至該服務的許可。

當名為 `marymajor` 的 IAM 使用者嘗試使用主控台在 Amazon MQ 中執行動作時，發生下列範例錯誤。但是，動作請求服務具備服務角色授予的許可。Mary 沒有將角色傳遞至該服務的許可。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在這種情況下，Mary 的政策必須更新，允許她執行 `iam:PassRole` 動作。

如需任何協助，請聯絡您的 AWS 管理員。您的管理員提供您的登入憑證。

## 我想允許 AWS 帳戶外的人員存取我的 Amazon MQ 資源

您可以建立一個角色，讓其他帳戶中的使用者或您組織外部的人員存取您的資源。您可以指定要允許哪些信任對象取得該角色。針對支援基於資源的政策或存取控制清單 (ACL) 的服務，您可以使用那些政策來授予人員存取您資源的許可。

若要進一步了解，請參閱以下內容：

- 若要了解 Amazon MQ 是否支援這些功能，請參閱 Amazon MQ 如何搭配 IAM 運作。
- 若要了解如何存取您擁有的所有 AWS 帳戶 所提供的資源，請參閱《IAM 使用者指南》中的將存取權提供給您所擁有的另一個 AWS 帳戶 中的 IAM 使用者。
- 若要了解如何將資源的存取權提供給第三方 AWS 帳戶，請參閱《IAM 使用者指南》中的將存取權提供給第三方擁有的 AWS 帳戶。
- 若要了解如何透過聯合身分提供存取權，請參閱《IAM 使用者指南》中的將存取權提供給在外部進行身分驗證的使用者 (聯合身分)。
- 若要了解使用角色和資源型政策進行跨帳戶存取之間的差異，請參閱《IAM 使用者指南》中的 IAM 角色與資源型政策的差異。

# Amazon MQ 的合規驗證

在多個 AWS 合規計劃中，第三方稽核人員會評估 Amazon MQ 的安全與合規。這些包括 SOC、PCI、HIPAA 等。

要了解 AWS 服務 是否在特定合規計畫範圍內，請參閱合規計畫範圍內的 AWS 服務，並選擇您感興趣的合規計畫。如需一般資訊，請參閱 AWS 法規遵循方案。

您可使用 AWS Artifact 下載第三方稽核報告。如需詳細資訊，請參閱 AWS Artifact 中的下載報告。

您使用 AWS 服務 時的法規遵循責任取決於資料的敏感度、您的公司的合規目標，以及適用的法律和法規。AWS 提供以下資源協助您處理法規遵循事宜：

- 安全與合規快速入門指南 – 這些部署指南討論在 AWS 上部署以安全及合規為重心的基準環境的架構考量和步驟。
- Amazon Web Services 的 HIPAA 安全與法規遵循架構：本白皮書說明公司可如何運用 AWS 來建立符合 HIPAA 規定的應用程式。

> **Note**
>
> 並非全部的 AWS 服務 都符合 HIPAA 資格。如需詳細資訊，請參閱 HIPAA 資格服務參照。

- AWS 法規遵循資源：這組手冊和指南可能適用於您的產業和位置。

- AWS 客戶合規指南：透過合規角度瞭解共同的責任模式。這份指南橫跨多個架構 (包含國家標準技術研究所 (NIST)、支付卡產業安全標準委員會 (PCI) 和國際標準組織 (ISO))，總結保護 AWS 服務的最佳實務並將指導方針對應至安全控制。
- AWS Config 開發人員指南中的使用規則評估資源：AWS Config 服務可評估資源組態對於內部實務、業界準則和法規的合規狀態。
- AWS Security Hub – 此 AWS 服務 可供您全面檢視 AWS 中的安全狀態。Security Hub 使用安全控制，可評估您的 AWS 資源並檢查您的法規遵循是否符合安全業界標準和最佳實務。如需支援的服務和控制清單，請參閱 Security Hub controls reference。
- AWS Audit Manager – 此 AWS 服務 可協助您持續稽核 AWS 使用情況，以簡化管理風險與法規與業界標準的法規遵循方式。

# Amazon MQ 的恢復能力

AWS 全球基礎設施是以 AWS 區域與可用區域為中心建置的。AWS區域提供多個分開且隔離的實際可用區域，它們以低延遲、高輸送量和高度備援聯網功能相互連結。透過可用區域，您可以設計與操作的應用程式和資料庫，在可用區域之間自動容錯移轉而不會發生中斷。可用區域的可用性、容錯能力和擴充能力，均較單一或多個資料中心的傳統基礎設施還高。

如需 AWS 區域與可用區域的詳細資訊，請參閱 AWS 全球基礎設施。

# Amazon MQ 的基礎設施安全性

是一項受管服務，受 AWS 全球網路安全的保護。如需有關 AWS 安全服務以及 AWS 如何保護基礎設施的詳細資訊，請參閱 AWS 雲端安全。若要使用基礎設施安全性的最佳實務來設計您的 AWS 環境，請參閱安全性支柱 AWS 架構良好的框架中的基礎設施保護。

您可使用 AWS 發佈的 API 呼叫，透過網路存取。用戶端必須支援下列項目：

- Transport Layer Security (TLS)。我們需要 TLS 1.2 並建議使用 TLS 1.3。
- 具備完美轉送私密 (PFS) 的密碼套件，例如 DHE (Ephemeral Diffie-Hellman) 或 ECDHE (Elliptic Curve Ephemeral Diffie-Hellman)。現代系統 (如 Java 7 和更新版本) 大多會支援這些模式。

此外，請求必須使用存取索引鍵 ID 和與 IAM 主體相關聯的私密存取索引鍵來簽署。或者，您可以使用 AWS Security Token Service (AWS STS) 來產生暫時安全憑證來簽署請求。

# Amazon MQ 的安全最佳實務

以下設計模式可以改善 Amazon MQ 代理程式的安全性。

主題



如需 Amazon MQ 如何加密資料的詳細資訊，以及支援的通訊協定清單，請參閱資料保護。

## 偏好無法公開存取的代理程式

建立的代理程式若無法公開存取，則您無法從 VPC 外存取它們。這可讓您的代理程式更不易受到來自公有網際網路的分散式阻斷服務 (DDoS) 攻擊。如需詳細資訊，請參閱本指南中的 在沒有公開存取性的情況下存取代理程式 Web 主控台 以及 AWS 安全部落格上的如何藉由減少攻擊表面協助針對 DDoS 攻擊做準備。

## 一律設定授權映射

因為 ActiveMQ 沒有根據預設來設定的任何授權映射，所以任何已驗證的使用者都可對代理程式執行任何動作。因此，最佳實務為依群組來限制許可。如需詳細資訊，請參閱 `authorizationEntry`。

> **Important**
>
> 如果您指定的授權映射不包含 `activemq-webconsole` 群組，您便無法使用 ActiveMQ Web 主控台，因為該群組未獲授權傳送或接收來自 Amazon MQ 代理程式的訊息。

## 透過 VPC 安全群組封鎖不必要的通訊協定

為了改善安全性，您應該正確設定您的 Amazon VPC 安全群組，限制與不必要通訊協定和連接埠的連線。例如，若要同時限制大多數通訊協定的存取，又能存取 OpenWire 和 Web 主控台，您可以僅開放存取 61617 和 8162。這樣做可封鎖非使用中的通訊協定，且同時允許 OpenWire 和 Web 主控台正常運作，限制您的公開情況。

僅允許您正在使用的通訊協定連接埠。

- AMQP: 5671
- MQTT: 8883
- OpenWire: 61617
- STOMP: 61614
- WebSocket: 61619

如需更多詳細資訊，請參閱：

- Configure Additional Broker Settings
- VPC 的安全群組
- VPC 的預設安全群組
- 使用安全群組

# 記錄和監控 Amazon MQ 代理程式

監控是維護您 AWS 解決方案之可靠性、可用性和效能的重要部分。您應該從 AWS 解決方案各個部分收集監控資料，以便在發生多點失敗時，可更輕鬆地偵錯。AWS 提供多種工具，讓您監控 Amazon MQ 資源及回應潛在的事件：

主題



## 存取 Amazon MQ 的 CloudWatch 指標

Amazon MQ 與 Amazon CloudWatch 會整合在一起，以便您可使用 CloudWatch 為您的 ActiveMQ 代理程式及代理程式的目標 (佇列及主題) 檢視及分析指標。您可以從 CloudWatch 主控台、AWS CLI 或 CloudWatch CLI 檢視及分析 Amazon MQ 指標。Amazon MQ 的 CloudWatch 指標會每分鐘自動從代理程式輪詢並推送到 CloudWatch。

如需 Amazon MQ 指標的完整清單，請參閱 Monitoring Amazon MQ using CloudWatch。

如需針對指標建立 CloudWatch 警示的相關資訊，請參閱 Amazon CloudWatch 使用者指南中的建立或編輯 CloudWatch 警示。

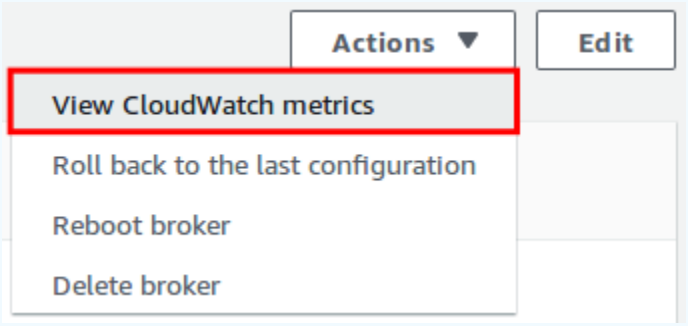
> **Note**
>
> Amazon MQ 指標是免費的並在 CloudWatch 中回報。系統會以 Amazon MQ 服務的一部分來提供這些指標。
> 對於 ActiveMQ 代理程式，CloudWatch 只會監控前 1000 個目的地。
> 對於 RabbitMQ 代理程式，CloudWatch 只會監控前 500 個目的地 (依消費者數量排序)。

主題

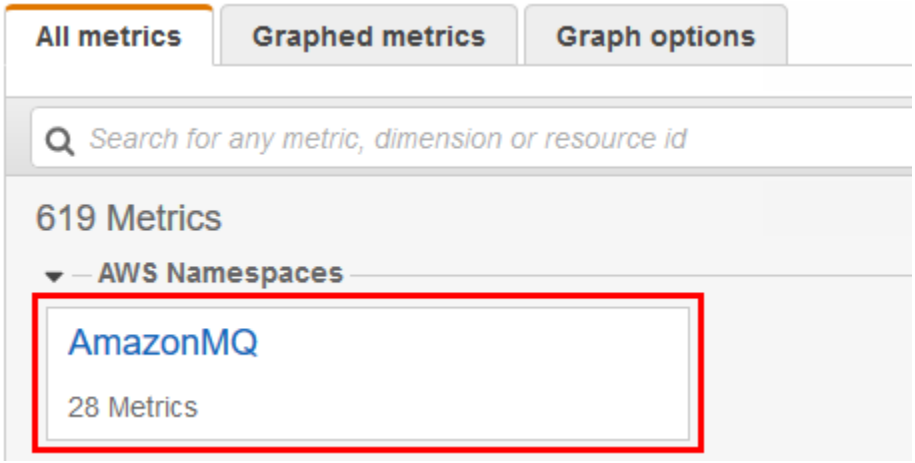
- Amazon CloudWatch API

## AWS Management Console

以下範例顯示如何使用 AWS Management Console 存取 Amazon MQ 的 CloudWatch 指標。

> **Note**
>
> 如果您已經登入 Amazon MQ 主控台，請在代理程式 Details (詳細資訊) 頁面上，選擇 Actions (動作)、View CloudWatch metrics (檢視 CloudWatch 指標)。
>
> 
> 

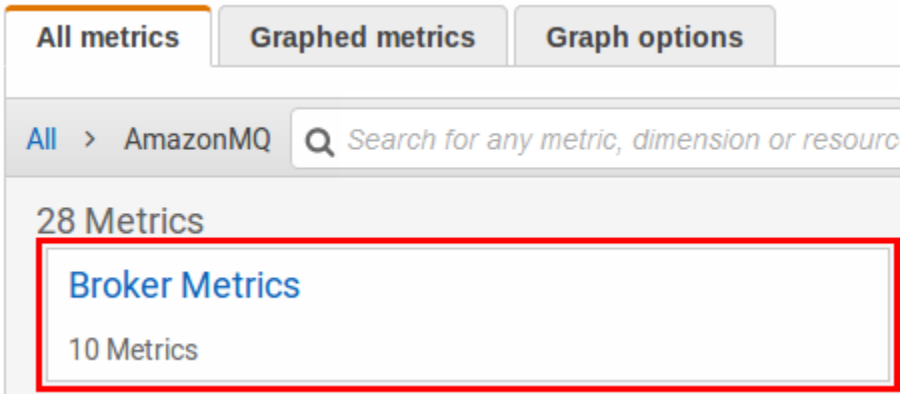
1. 登入 CloudWatch 主控台。
2. 在導覽面板上，選擇 Metrics (指標)。
3. 選取 AmazonMQ 指標命名空間。




4. 選取以下其中一個指標維度：
   - Broker Metrics (中介裝置指標)
   - Queue Metrics by Broker (依照中介裝置的佇列指標)

- Topic Metrics by Broker (依照中介裝置的主題指標)

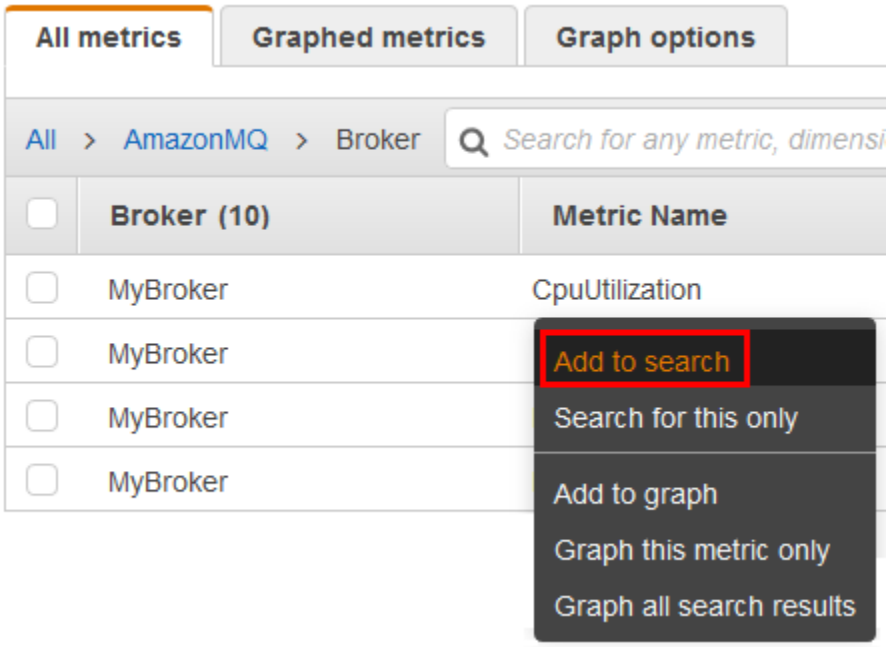
在此範例中，會選取 Broker Metrics (代理程式指標)。




5. 您現在可以檢查 Amazon MQ 指標：

- 若要排序指標，請使用欄標題。

- 若要將指標圖形化，請選取指標旁的核取方塊。
- 若要依指標篩選，請選擇指標名稱，然後選擇 Add to search (新增至搜尋)。



### AWS Command Line Interface

若要使用 AWS CLI 存取 Amazon MQ 指標，請使用 `get-metric-statistics` 命令。

如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南中的取得指標的統計資料。

### Amazon CloudWatch API

若要使用 CloudWatch API 存取 Amazon MQ 指標，請使用 `GetMetricStatistics` 動作。

如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南中的取得指標的統計資料。

## 使用 Amazon CloudWatch 監控 Amazon MQ 代理程式

Amazon MQ 與 Amazon CloudWatch 會整合在一起，以便您可使用 CloudWatch 為您的 ActiveMQ 代理程式及代理程式的目標 (佇列及主題) 檢視及分析指標。您可以從 CloudWatch 主控台、AWS CLI 或 CloudWatch CLI 檢視及分析 Amazon MQ 指標。Amazon MQ 的 CloudWatch 指標會每分鐘自動從代理程式輪詢並推送到 CloudWatch。

如需相關資訊，請參閱存取 Amazon MQ 的 CloudWatch 指標。

> **Note**
>
> 以下統計資料適用於所有指標：
>
> - `Average`
> - `Minimum`
> - `Maximum`
> - `Sum`

`AWS/AmazonMQ` 命名空間包含下列指標。

主題

- 記錄和監控 Amazon MQ for ActiveMQ 代理程式
- 記錄和監控 Amazon MQ for RabbitMQ 代理程式

# 記錄和監控 Amazon MQ for ActiveMQ 代理程式

## Amazon MQ for ActiveMQ 指標

| 指標 | 單位 | 描述 |
| --- | --- | --- |
| `AmqpMaximumConnections` | 計數 | 您可以使用 AMQP 連線至代理程式的最大用戶端數量。如需連線配額的詳細資訊，請參閱 Quotas in Amazon MQ。 |
| `BurstBalance` | 百分比 | Amazon EBS 磁碟區上剩餘的爆量額度百分比，可用來保存訊息資料，以用於輸送量最佳化代理程式。如果此餘額達到零，Amazon EBS 磁碟區提供的 IOPS 會減少，直到爆量餘額重新填滿為止。如需 Amazon EBS 中高載餘額運作方式的詳細資訊，請參閱：輸入/輸出額度和高載效能。 |
| `CpuCreditBalance` | 額度 (vCPU-分鐘) | **Important**<br>這個指標僅適用於 `mq.t2.micro` 代理程式執行個體類型。CPU 額度指標僅提供 5 分鐘間隔。<br><br>自執行個體啟動或開始後，累積獲得的 CPU 點數數量 (包括啟動額度數量)。額度餘額可供代理程式執行個體為超越基準 CPU 使用率的大幅提升支付費用。 |

| 指標 | 單位 | 描述 |
| --- | --- | --- |
| | | 獲得額度後，額度會在額度餘額中累積，並在支付額度之後，從額度餘額中移出。額度餘額已達上限。當到達限額之後，所有獲得的新額度都將被捨棄。 |
| CpuUtilization | 百分比 | 代理程式目前使用的已配置 Amazon EC2 運算單位的百分比。 |
| CurrentConnectionsCount | 計數 | 目前在代理程式上的現有連線數量。 |
| EstablishedConnectionsCount | 計數 | 已建置於代理程式上的作用中及非作用中連線總數量。 |
| HeapUsage | 百分比 | 代理程式目前使用 ActiveMQ JVM 記憶體限制。 |
| InactiveDurableTopicSubscribersCount | 計數 | 非作用中耐用性主題訂閱者人數，最多可達 2000 人。 |
| JobSchedulerStorePercentUsage | 百分比 | 任務排程器存放區使用的磁碟空間百分比。 |
| JournalFilesForFastRecovery | 計數 | 將在正常關機後重播的日誌檔案數量。 |
| JournalFilesForFullRecovery | 計數 | 將在非正常關機後重播的日誌檔案數量。 |
| MqttMaximumConnections | 計數 | 您可以使用 MQTT 連線至代理程式的最大用戶端數量。如需連線配額的詳細資訊，請參閱 Quotas in Amazon MQ。 |

| 指標 | 單位 | 描述 |
| --- | --- | --- |
| NetworkConnectorConnectionCount | 計數 | 使用 NetworkConnector 在代理程式網路中連線至代理程式的節點數。 |
| NetworkIn | 位元組 | 代理程式的傳入流量。 |
| NetworkOut | 位元組 | 代理程式的傳出流量。 |
| OpenTransactionCount | 計數 | 進行中的交易總數。 |
| OpenwireMaximumConnections | 計數 | 您可以使用 OpenWire 連線至代理程式的最大用戶端數量。如需連線配額的詳細資訊，請參閱 Quotas in Amazon MQ。 |
| StompMaximumConnections | 計數 | 您可以使用 STOMP 連線至代理程式的最大用戶端數量。如需連線配額的詳細資訊，請參閱 Quotas in Amazon MQ。 |
| StorePercentUsage | 百分比 | 儲存限制所用的百分比符號。如果這個數字達到 100，代理程式將會拒絕訊息。 |
| TempPercentUsage | 百分比 | 非持久性訊息使用的可用臨時儲存百分比。 |
| TotalConsumerCount | 計數 | 消費者向目前代理程式目的地訂閱的訊息數量。 |
| TotalMessageCount | 計數 | 儲存在代理程式上的訊息數目。 |
| TotalProducerCount | 計數 | 生產者在目前代理程式目的地啟用的訊息數量。 |

| 指標 | 單位 | 描述 |
| --- | --- | --- |
| VolumeReadOps | 計數 | 在 Amazon EBS 磁碟區上執行的讀取操作數目。 |
| VolumeWriteOps | 計數 | 在 Amazon EBS 磁碟區上執行的寫入操作數目。 |
| WsMaximumConnections | 計數 | 您可以使用 WebSocket 連線至代理程式的最大用戶端數量。如需連線配額的詳細資訊，請參閱 Quotas in Amazon MQ。 |

ActiveMQ 代理程式指標的維度

| 維度 | 描述 |
| --- | --- |
| Broker | 代理程式的名稱<br><br>**Note**<br>單一執行個體代理程式有尾碼 -1。符合高可用性的作用中/待命代理程式，包含可代表其備援組合的尾碼 -1 和 -2。 |

## ActiveMQ 目的地 (佇列和主題) 指標

**Important**

下列指標包含 CloudWatch 輪詢期間適用的每分鐘計數。

- EnqueueCount
- ExpiredCount
- DequeueCount
- DispatchCount

- InFlightCount

例如，在五分鐘 CloudWatch 期間 中，EnqueueCount 有五個計數值，每個計數值為期間的一分鐘部分。Minimum 和 Maximum 統計資料會提供在該期間的最低和最高每分鐘值。

| 指標 | 單位 | 描述 |
| --- | --- | --- |
| ConsumerCount | 計數 | 訂閱目標的使用者數量。 |
| EnqueueCount | 計數 | 每分鐘發送到目的地的訊息數量。 |
| EnqueueTime | 時間(毫秒) | 從訊息送達代理程式起至其傳送給消費者這段期間的端對端延遲。<br><br>**Note**<br>EnqueueTime 不會測量從生產者傳送訊息直到送達代理程式為止的端對端延遲，也不會測量從代理程式接收訊息直到代理程式認可訊息為止的延遲。相反，EnqueueTime 是從代理程式收到訊息那一刻直到訊息成功傳送給消費者為止的毫秒數。 |
| ExpiredCount | 計數 | 每分鐘因訊息過期而無法傳送的訊息數量。 |

| 指標 | 單位 | 描述 |
| --- | --- | --- |
| DispatchCount | 計數 | 每分鐘發送給消費者的訊息數量。 |
| DequeueCount | 計數 | 每分鐘經消費者認可的訊息數量。 |
| InFlightCount | 計數 | 傳送到未獲得認可之消費者的訊息數量。 |
| ReceiveCount | 計數 | 已從遠端代理程式收到有關雙工連接器的訊息數量。 |
| MemoryUsage | 百分比 | 目標位置當前使用的內存限制的百分比。 |
| ProducerCount | 計數 | 目標位置的創建者數量。 |
| QueueSize | 計數 | 佇列中的訊息數量。<br><br>**Important**<br>此指標僅適用於隊列。 |
| TotalEnqueueCount | 計數 | 已傳送給代理程式的訊息總數。 |
| TotalDequeueCount | 計數 | 用戶端已使用的訊息總數。 |

**Note**

TotalEnqueueCount 和 TotalDequeueCount 指標包含諮詢主題的訊息。如需有關諮詢主題訊息的詳細資訊，請參閱 ActiveMQ 文件。

ActiveMQ 目的地 (佇列和主題) 指標的維度

| 維度 | 描述 |
| --- | --- |
| Broker | 代理程式的名稱。<br><br>**Note**<br>單一執行個體代理程式有尾碼 -1。符合高可用性的作用中/待命代理程式，包含可代表其備援組合的尾碼 -1 和 -2。 |
| Topic 或 Queue | 主題或佇列的名稱。 |
| NetworkConnector | 網路連接器的名稱。 |

# 記錄和監控 Amazon MQ for RabbitMQ 代理程式

## RabbitMQ 代理程式指標

| 指標 | 單位 | 描述 |
| --- | --- | --- |
| ExchangeCount | 計數 | 代理程式上設定的交換總數。 |
| QueueCount | 計數 | 代理程式上設定的佇列總數。 |
| ConnectionCount | 計數 | 建立於代理程式上的連線總數。 |
| ChannelCount | 計數 | 建立於代理程式上的通道總數。 |
| ConsumerCount | 計數 | 連線至代理程式的消費者總數。 |
| MessageCount | 計數 | 佇列中的訊息總數。 |

| 指標 | 單位 | 描述 |
| --- | --- | --- |
| | | **Note**<br>產生的數字是代理程式上準備就緒和未確認的訊息總和。 |
| MessageReadyCount | 計數 | 佇列中準備就緒的訊息總數。 |
| MessageUnacknowledgedCount | 計數 | 佇列中未認可的訊息總數。 |
| PublishRate | 計數 | 訊息發佈至代理程式的速率。<br>產生的數字代表取樣時每秒的訊息數目。 |
| ConfirmRate | 計數 | RabbitMQ 伺服器確認已發佈訊息的速率。您可將此指標與 PublishRate 比較，更進一步了解您的代理程式效能。<br>產生的數字代表取樣時每秒的訊息數目。 |
| AckRate | 計數 | 消費者認可訊息的速率。<br>產生的數字代表取樣時每秒的訊息數目。 |
| SystemCpuUtilization | 百分比 | 代理程式目前使用的已配置 Amazon EC2 運算單位的百分比。對於叢集部署，此值表示所有三個 RabbitMQ 節點之對應指標值的彙總。 |

| 指標 | 單位 | 描述 |
| --- | --- | --- |
| RabbitMQMemLimit | 位元組 | RabbitMQ 代理程式的 RAM 限制。對於叢集部署，此值表示所有三個 RabbitMQ 節點之對應指標值的彙總。 |
| RabbitMQMemUsed | 位元組 | RabbitMQ 代理程式所使用的 RAM 磁碟區。對於叢集部署，此值表示所有三個 RabbitMQ 節點之對應指標值的彙總。 |
| RabbitMQDiskFreeLimit | 位元組 | RabbitMQ 代理程式的磁碟限制。對於叢集部署，此值表示所有三個 RabbitMQ 節點之對應指標值的彙總。每個執行個體大小的此指標都不同。如需 Amazon MQ 執行個體類型的詳細資訊，請參閱 the section called “Amazon MQ for RabbitMQ 執行個體類型”。 |
| RabbitMQDiskFree | 位元組 | RabbitMQ 代理程式中可用的可用磁碟空間總數量。當磁碟使用量超過限制時，叢集會封鎖所有的生產者連線。對於叢集部署，此值表示所有三個 RabbitMQ 節點之對應指標值的彙總。 |
| RabbitMQFdUsed | 計數 | 使用的檔案描述項數目。對於叢集部署，此值表示所有三個 RabbitMQ 節點之對應指標值的彙總。 |
| RabbitMQIOReadAverageTime | 計數 | RabbitMQ 執行一次讀取操作的平均時間 (毫秒)。該值與訊息大小成正比。 |

| 指標 | 單位 | 描述 |
| --- | --- | --- |
| RabbitMQIOWriteAverageTime | 計數 | RabbitMQ 執行一次寫入操作的平均時間 (毫秒)。該值與訊息大小成正比。 |

## RabbitMQ 代理程式指標的維度

| 維度 | 描述 |
| --- | --- |
| Broker | 代理程式的名稱。 |

## RabbitMQ 節點指標

| 指標 | 單位 | 描述 |
| --- | --- | --- |
| SystemCpuUtilization | 百分比 | 代理程式目前使用的已配置 Amazon EC2 運算單位的百分比。 |
| RabbitMQMemLimit | 位元組 | RabbitMQ 節點的 RAM 限制。 |
| RabbitMQMemUsed | 位元組 | RabbitMQ 節點所使用的 RAM 磁碟區。當記憶體使用超過限制時，叢集將封鎖所有的生產者連線。 |
| RabbitMQDiskFreeLimit | 位元組 | RabbitMQ 節點的磁碟限制。每個執行個體大小的此指標都不同。如需 Amazon MQ 執行個體類型的詳細資訊，請參閱 the section called “Amazon MQ for RabbitMQ 執行個體類型”。 |

| 指標 | 單位 | 描述 |
| --- | --- | --- |
| RabbitMQDiskFree | 位元組 | RabbitMQ 節點中可用的可用磁碟空間總數量。當磁碟使用量超過限制時，叢集會封鎖所有的生產者連線。 |
| RabbitMQFdUsed | 計數 | 使用的檔案描述項數目。 |

## RabbitMQ 節點指標的維度

| 維度 | 描述 |
| --- | --- |
| Node | 節點的名稱。<br><br>**Note**<br>節點名稱包含兩個部分：前置詞 (通常為 rabbit) 和主機名稱。例如：rabbit@ip-10-0-0-230.us-west-2.compute.internal 是節點名稱，其前置詞 rabbit 和主機名稱 ip-10-0-0-230.us-west-2.compute.internal 。 |
| Broker | 代理程式的名稱。 |

## RabbitMQ 佇列指標

| 指標 | 單位 | 描述 |
| --- | --- | --- |
| ConsumerCount | 計數 | 訂閱佇列的消費者數目。 |
| MessageReadyCount | 計數 | 目前可傳送的訊息數目。 |

| 指標 | 單位 | 描述 |
| --- | --- | --- |
| MessageUnacknowledgedCount | 計數 | 伺服器正在等待認可的訊息數目。 |
| MessageCount | 計數 | MessageReadyCount 和 MessageUnacknowledgedCount 的總數 (也稱為佇列深度)。 |

### RabbitMQ 佇列指標的維度

> **Note**
>
> Amazon MQ for RabbitMQ 不會為名稱包含空格、定位字元或其他非 ASCII 字元的虛擬主機和佇列發佈指標。
> 如需維度名稱的詳細資訊，請參閱《Amazon CloudWatch API 參考》中的維度。

| 維度 | 描述 |
| --- | --- |
| Queue | 佇列的名稱。 |
| VirtualHost | 虛擬主機的名稱。 |
| Broker | 代理程式的名稱。 |

## 使用 AWS CloudTrail 記錄 Amazon MQ API 呼叫

Amazon MQ 已與 AWS CloudTrail 整合，此服務會提供 Amazon MQ 呼叫的記錄，而這些呼叫是由使用者、角色或 AWS 服務所發出。CloudTrail 會將與 Amazon MQ 代理程式和組態相關的 API 呼叫擷取為事件，包括來自 Amazon MQ 主控台的呼叫以及來 Amazon MQ API 的程式碼呼叫。如需有關 CloudTrail 的詳細資訊，請參閱 AWS CloudTrail 使用者指南。

> **Note**
>
> CloudTrail 不會記錄與 ActiveMQ 操作 (例如，傳送和接收訊息) 相關的 API 呼叫，或與 ActiveMQ Web 主控台相關的 API 呼叫。若要記錄與 ActiveMQ 操作相關的資訊，您可以設定 Amazon MQ 以將一般和稽核日誌發佈至 Amazon CloudWatch Logs。

您可以使用 CloudTrail 收集的資訊，識別對於 Amazon MQ API 的特定請求、申請者的 IP 地址、申請者的身分、請求的日期和時間等。如果您建立追蹤，就可以將 CloudTrail 事件持續傳送到 Amazon S3 儲存貯體。如果未設定追蹤，您可以在 CloudTrail 主控台的事件歷史記錄中檢視最新的事件。如需詳細資訊，請參閱《AWS CloudTrail 使用者指南》中的建立追蹤的概觀。

## CloudTrail 中的 Amazon MQ 資訊

當您建立 AWS 帳戶時，CloudTrail 便已啟用。支援的 Amazon MQ 事件發生時會記錄在 CloudTrail 事件中，連同其他 AWS 服務事件在一起記錄在事件歷程記錄中。您可以檢視、搜尋和下載 AWS 帳戶的最新事件。如需詳細資訊，請參閱《AWS CloudTrail 使用者指南》中的使用 CloudTrail 事件歷史記錄檢視事件。

追蹤可讓 CloudTrail 將日誌檔案傳送至 Amazon S3 儲存貯體。您可以建立追蹤來持續記錄 AWS 帳戶中的事件。依預設，當您使用 AWS Management Console建立追蹤時，該追蹤會套用到所有 AWS 區域。該追蹤會記錄來自所有 AWS 區域的事件，並將日誌檔案傳送到您指定的 Amazon S3 儲存貯體。您可配置其他 AWS 服務，進一步分析和處理 CloudTrail 記錄中所收集的事件資料。如需詳細資訊，請參閱《AWS CloudTrail 使用者指南》中的以下主題：

- CloudTrail 支援的服務和整合
- 設定 CloudTrail 的 Amazon SNS 通知
- 從多個區域接收 CloudTrail 日誌檔案
- 從多個帳戶接收 CloudTrail 日誌檔案

Amazon MQ 支援將下列 API 的請求參數和回應同時記錄為 CloudTrail 日誌檔中的事件：

- `CreateConfiguration`
- `DeleteBroker`
- `DeleteUser`
- `RebootBroker`

- UpdateBroker

> **Note**
>
> 當您重新啟動代理程式時，系統會記錄 RebootBroker 日誌檔案。在維護時段的期間，服務會自動重新啟動，而且不會記錄 BootBroker 日誌檔案。

> **Important**
>
> 對於以下 API 的 GET 方法，系統會記錄請求參數，但會修訂回應：
>
> - DescribeBroker
> - DescribeConfiguration
> - DescribeConfigurationRevision
> - DescribeUser
> - ListBrokers
> - ListConfigurationRevisions
> - ListConfigurations
> - ListUsers
>
> 對於以下 API，data 和 password 請求參數會透過星號 (****) 隱藏：
>
> - CreateBroker (POST)
> - CreateUser (POST)
> - UpdateConfiguration (PUT)
> - UpdateUser (PUT)

每一個事件或記錄項目都包含申請者的相關資訊。此資訊可協助您判斷下列事項：

- 該請求是否以根登入資料或 使用者登入資料提出？
- 該請求是否以角色或聯合身分使用者的暫時安全登入資料提出？
- 該請求是否由其他 AWS 帳戶提出？

如需詳細資訊，請參閱《AWS CloudTrail 使用者指南》中的 CloudTrail userIdentity 元素。

## 範例：Amazon MQ 日誌檔案項目

追蹤是一種組態，可讓事件以日誌檔案的形式傳送到指定的 Amazon S3 儲存貯體。CloudTrail 日誌檔案包含一或多個日誌項目。

事件代表任何來源的單一請求，並包含下列項目的相關資訊：對 Amazon MQ API 的請求、請求者的 IP 地址、請求者的身分、請求的日期和時間等等。

以下範例顯示 `CreateBroker` API 呼叫的 CloudTrail 日誌項目。

> **Note**
>
> 因為 CloudTrail 日誌檔案不是公有 API 的已排序堆疊追蹤，因此不會以任何特定順序列出資訊。

```
{
    "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"
}
```

# 設定 Amazon MQ 將日誌發佈到 Amazon CloudWatch Logs

Amazon MQ 與 Amazon CloudWatch Logs 整合，後者是從各種來源監控、存放及存取日誌檔的服務。例如，您可以設定 CloudWatch 警示，以接收代理程式重新啟動的通知，或排除 ActiveMQ 代理程式組態錯誤。如需 CloudWatch Logs 的詳細資訊，請參閱 Amazon CloudWatch Logs 使用者指南。

主題



## 設定 Amazon MQ for ActiveMQ 日誌

若要允許 Amazon MQ 將日誌發佈至 CloudWatch Logs，您必須將許可新增至 Amazon MQ 使用者，並在建立或重新啟動代理程式前，為 Amazon MQ 設定以資源為基礎的政策。

以下說明是為 ActiveMQ 代理程式設定 CloudWatch Logs 的步驟。

主題



### 了解 CloudWatch Logs 中的記錄結構

當您設定進階代理程式設定、當您建立代理程式，或當您編輯代理程式時，您可以啟用一般和稽核記錄。

一般記錄會啟用預設 `INFO` 記錄層級 (不支援 `DEBUG` 記錄)，並將 `activemq.log` 發佈到 CloudWatch 帳戶中的日誌群組。日誌群組具有如下的格式：

```
/aws/amazonmq/broker/b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9/general
```

稽核日誌可讓系統記錄使用 JMX 或使用 ActiveMQ Web 主控台所採取的管理動作，並將 `audit.log` 發佈到 CloudWatch 帳戶中的日誌群組。日誌群組具有如下的格式：

```
/aws/amazonmq/broker/b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9/audit
```

根據您具有單一執行個體代理程式，還是作用中/待命代理程式，Amazon MQ 會在每個日誌群組內建立一或兩個日誌串流。日誌串流具有如下的格式。

```
activemq-b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.log
activemq-b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-2.log
```

-1 和 -2 尾碼表示個別的代理程式執行個體。如需詳細資訊，請參閱《Amazon CloudWatch Logs 使用者指南》中的使用日誌群組和日誌串流。

## 將 **CreateLogGroup** 許可新增至 Amazon MQ 使用者

若要允許 Amazon MQ 建立 CloudWatch Logs 日誌群組，您必須確保建立或重新啟動代理程式的 IAM 使用者具有 `logs:CreateLogGroup` 許可。

> **Important**
>
> 在使用者建立或重新啟動代理程式之前，如果您未將 `CreateLogGroup` 許可新增至 Amazon MQ 使用者，則 Amazon MQ 不會建立日誌群組。

下列範例 以 IAM 為基礎的政策將 `logs:CreateLogGroup` 的許可授予此政策附加至的使用者。

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": "logs:CreateLogGroup",
         "Resource": "arn:aws:logs:*:*:log-group:/aws/amazonmq/*"
      }
   ]
}
```

> **Note**
>
> 在此，使用者一詞是指使用者，而不是 Amazon MQ 使用者，後者是在設定新代理程式時建立的使用者。如需有關設定使用者和設定 IAM 政策的詳細資訊，請參閱《IAM 使用者指南》中的身分管理概觀一節。

如需詳細資訊，請參閱 Amazon CloudWatch Logs API 參考中的 `CreateLogGroup`。

## 為 Amazon MQ 設定資源型政策。

> **Important**
>
> 如果您未對 Amazon MQ 設定資源型政策，則代理程式無法將日誌發佈到 CloudWatch Logs。

若要允許 Amazon MQ 將日誌發佈到 CloudWatch Logs 日誌群組，請設定資源型政策，提供 Amazon MQ 存取以下 CloudWatch Logs API 動作的權限：

- `CreateLogStream` – 為指定的日誌群組建立 CloudWatch Logs 日誌串流。
- `PutLogEvents` – 將事件傳送到指定的 CloudWatch Logs 日誌串流。

以下以資源為基礎的政策會將 `logs:CreateLogStream` 和 `logs:PutLogEvents` 的許可授予 AWS。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": { "Service": "mq.amazonaws.com" },
            "Action": [ "logs:CreateLogStream", "logs:PutLogEvents" ],
            "Resource": "arn:aws:logs:*:*:log-group:/aws/amazonmq/*"
        }
    ]
}
```

這種以資源為基礎的政策必須使用 AWS CLI 進行設定，如下列命令所示。在此範例中，以自己的資訊取代 *us-east-1*。

```
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**
>
> 由於這個範例使用 /aws/amazonmq/ 字首，您需要在每個區域僅對每個 AWS 帳戶設定資源型政策一次。

## 預防跨服務混淆代理人

混淆代理人問題屬於安全性議題，其中沒有執行動作許可的實體可以強制具有更多許可的實體執行該動作。在 AWS 中，跨服務模擬可能會導致混淆代理人問題。在某個服務 (呼叫服務) 呼叫另一個服務 (被呼叫服務) 時，可能會發生跨服務模擬。可以操縱呼叫服務來使用其許可，以其不應有存取許可的方式對其他客戶的資源採取動作。為了預防這種情況，AWS 提供的工具可協助您保護所有服務的資料，而這些服務主體已獲得您帳戶中資源的存取權。

我們建議您在 Amazon MQ 資源型政策中使用 `aws:SourceArn` 和 `aws:SourceAccount` 全域條件內容鍵，以限制 CloudWatch Logs 對一個或多個指定代理程式的存取。

> **Note**
>
> 如果同時使用全域條件內容索引鍵，則在相同政策陳述式中使用`aws:SourceAccount` 值和 `aws:SourceArn` 值中的帳戶時，必須使用相同的帳戶 ID。

以下範例示範了限制 CloudWatch Logs 對單一 Amazon MQ 代理程式之存取的資源型政策。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "mq.amazonaws.com"
```

```
      },
      "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "arn:aws:logs:*:*:log-group:/aws/amazonmq/*",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012",
          "aws:SourceArn": "arn:aws:mq:us-
east-2:123456789012:broker:MyBroker:b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9"
        }
      }
    }
  ]
}
```

您還可以設定資源型政策，以限制 CloudWatch Logs 對帳戶中所有代理程式的存取，如下所示。

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Principal": {
          "Service": [
            "mq.amazonaws.com"
          ]
        },
        "Action": [
          "logs:CreateLogStream",
          "logs:PutLogEvents"
        ],
        "Resource": "arn:aws:logs:*:*:log-group:/aws/amazonmq/*",
        "Condition": {
          "ArnLike": {
            "aws:SourceArn": "arn:aws:mq:*:123456789012:broker:*"
          },
          "StringEquals": {
            "aws:SourceAccount": "123456789012"
          }
        }
      }
```

```
    ]
  }
```

如需混淆代理安全問題的詳細資訊，請參閱《IAM 使用者指南》中的混淆代理問題。

### CloudWatch Logs 組態疑難排解

在某些情況下，CloudWatch Logs 可能無法總是按照預期行動。本節會提供常見問題的概觀，並說明如何解決問題。

日誌群組未出現在 CloudWatch 中

將 `CreateLogGroup` 許可新增至 Amazon MQ 使用者，然後重新啟動代理程式。如此便允許 Amazon MQ 建立日誌群組。

日誌串流未出現在 CloudWatch Logs 群組中

為 Amazon MQ 設定資源型政策。這可讓代理程式發佈其日誌。

## 設定 Amazon MQ for RabbitMQ 日誌

當您為 RabbitMQ 代理程式啟用 CloudWatch 記錄功能時，Amazon MQ 會使用服務連結的角色將一般日誌發佈到 CloudWatch。如果您第一次建立代理程式時沒有 Amazon MQ 服務連結的角色存在，Amazon MQ 會自動建立一個。所有後續的 RabbitMQ 代理程式都會使用相同的服務連結角色，將日誌發佈至 CloudWatch。

如需服務連結角色的詳細資訊，請參閱《AWS Identity and Access Management 使用者指南》中的使用服務連結角色。如需 Amazon MQ 如何使用服務連結角色的詳細資訊，請參閱 the section called “使用服務連結角色”。

# Amazon MQ 的配額

本主題列出 Amazon MQ 中的配額。您可以針對特定的 AWS 帳戶變更下列許多配額。若要請求提高限制，請參閱 Amazon Web Services 一般參考 中的 AWS Service Quotas。即使套用上限，更新的限制也不會顯示。如需檢視 Amazon CloudWatch 中目前連線限制的詳細資訊，請參閱使用 Amazon CloudWatch 監控 Amazon MQ 代理程式。

主題



## 中介裝置

下表列出與 Amazon MQ 代理程式相關的配額。

| 限制 | 描述 |
| --- | --- |
| 代理程式名稱 | • 在您的 AWS 帳戶中必須是唯一的。<br>• 長度必須介於 1 至 50 個字元之間。<br>• 必須僅包含 ASCII 可列印字元集中指定的字元。<br>• 只能包含英數字元、破折號、句點、底線和波狀符號 (- . _ ~)。 |
| 每區域的代理程式數目 | 50 |
| 較小型代理程式每個通訊協定的線路層級連線 | **Important**<br>不適用於 RabbitMQ 代理程式。 |

| 限制 | 描述 |
| --- | --- |
| | `mq.*.micro` 執行個體類型代理程式為 300 個。 |
| 較大型代理程式每個通訊協定的線路層級連線 | **Important**<br>不適用於 RabbitMQ 代理程式。<br><br>`mq.*.*large` 執行個體類型代理程式為 2,000 個。 |
| 每個代理程式的安全群組數 | 5 |
| CloudWatch 中監控的 ActiveMQ 目的地 (佇列與主題) | CloudWatch 只會監控前 1000 個目的地。 |
| 在 CloudWatch 中監控的 RabbitMQ 目的地 (佇列) | CloudWatch 只會監控前 500 個目的地 (依消費者數量排序)。 |
| 每個代理程式的標籤 | 50 |

## 組態

下表列出與 Amazon MQ 組態相關的配額。

**Important**
不適用於 RabbitMQ 代理程式。

| 限制 | 描述 |
| --- | --- |
| 組態名稱 | • 長度必須介於 1 至 150 個字元之間。<br>• |

| 限制 | 描述 |
| --- | --- |
| | 必須僅包含 ASCII 可列印字元集中指定的字元。<br>• 只能包含英數字元、破折號、句點、底線和波狀符號 (- . _ ~)。 |
| 每個組態的修訂數 | 300 |

# 使用者

下表列出與 Amazon MQ ActiveMQ 代理程式使用者相關的配額。

> **Important**
>
> 不適用於 RabbitMQ 代理程式。

| 限制 | 描述 |
| --- | --- |
| 使用者名稱 | • 長度必須介於 1 至 100 個字元之間。<br>• 必須僅包含 ASCII 可列印字元集中指定的字元。<br>• 只能包含英數字元、破折號、句點、底線和波狀符號 (- . _ ~)。<br>• 不得包含逗號 (,)。 |
| 密碼 | • 長度必須介於 12 至 250 個字元之間。<br>• 必須僅包含 ASCII 可列印字元集中指定的字元。 |

| 限制 | 描述 |
| --- | --- |
| | • 必須包含至少 4 個唯一字元。<br>• 不得包含逗號 (,)。 |
| 每個代理程式的使用者 (簡單身分驗證) | 250 |
| 每個使用者的群組 (簡單身分驗證) | 20 |

## 資料儲存體

下表列出與 Amazon MQ 資料儲存相關的配額。

| 限制 | 描述 |
| --- | --- |
| 每個較小代理程式的儲存容量 | `mq.*.micro` 執行個體類型代理程式為 20 GB。如需 Amazon MQ 執行個體類型的詳細資訊，請參閱 Broker instance types。 |
| 每個較大代理程式的儲存容量 | `mq.*.*large` 執行個體類型代理程式為 200 GB。如需 Amazon MQ 執行個體類型的詳細資訊，請參閱 Broker instance types。 |
| Amazon EBS 支援的每個代理程式的任務排程器使用量限制 | **Important**<br>不適用於 RabbitMQ 代理程式。<br><br>50 GB。如需任務排程器使用量的詳細資訊，請參閱 Apache ActiveMQ API 文件中的 `JobSchedulerUsage` 。 |
| 每個較小代理程式的暫時儲存容量。 | |

| 限制 | 描述 |
| --- | --- |
| | **Important**<br>不適用於 RabbitMQ 代理程式。<br><br>`mq.*.micro` 執行個體類型代理程式為 5 GB。 |
| 每個較大代理程式的暫時儲存容量。 | **Important**<br>不適用於 RabbitMQ 代理程式。<br><br>`mq.*.*large` 執行個體類型代理程式為 50 GB。 |

## API 調節

每個 AWS 帳戶的以下調節配額是跨所有 Amazon MQ API 而彙總的，以維護服務頻寬。如需 Amazon MQ API 的詳細資訊，請參閱 Amazon MQ REST API 參考。

**Important**
這些配額不適用於 Amazon MQ for ActiveMQ 或 Amazon MQ for RabbitMQ 代理程式傳訊 API。例如，Amazon MQ 不會調節訊息的傳送或接收。

| API 高載限制 | API 速率限制 |
| --- | --- |
| 100 | 15 |

# Amazon MQ 故障診斷

本節描述在使用 Amazon MQ 代理程式時可能遇到的常見問題，以及解決問題所需採取的步驟。

內容

- Amazon MQ for RabbitMQ：無效的 AWS Key Management Service 金鑰
  - 診斷和解決 INVALID_KMS_KEY
- Amazon MQ for ActiveMQ：已刪除彈性網路介面警示
- Amazon MQ for ActiveMQ：代理程式記憶體不足警示
- Amazon MQ for RabbitMQ：磁碟限制警示
  - 診斷和定址磁碟限制警示

# 疑難排解：一般

請使用此節中的資訊，協助診斷在使用 Amazon MQ 代理程式時可能遇到的常見問題，例如連線到代理程式的問題，以及代理程式重新啟動。

內容

- 我無法連線至代理程式 Web 主控台或端點。
- 我的代理程式正在執行，我可以使用 telnet 驗證連線能力，但我的用戶端無法連線並傳回 SSL 例外狀況。
- 我建立了代理程式，但代理程式建立失敗。
- 我的代理程式重新啟動，但我不確定原因。

## 我無法連線至代理程式 Web 主控台或端點。

如果您在使用 Web 主控台或線路層級端點連線至代理程式時遇到問題，建議您執行下列步驟。

1. 檢查您是否嘗試從防火牆後面連線到代理程式。您可能需要將防火牆設定為允許存取代理程式。
2. 檢查您是否嘗試使用 FIPS 端點連線到代理程式。Amazon MQ 僅在使用 API 操作時支援 FIPS 端點，而不支援與代理程式執行個體本身的線路連線。
3. 檢查 Public Accessibility (公開存取性) 選項是否設定為 Yes (是)。如果此選項設定為 No (否)，請檢查子網路的網路存取控制清單 (ACL)規則。如果您已建立自訂網路 ACL，您可能需要變更網路 ACL 規則，以提供代理程式的存取權。如需 Amazon VPC 聯網的詳細資訊，請參閱《Amazon VPC 使用者指南》中的啟用網際網路存取。
4. 檢查代理程式的安全群組規則。請確定您允許連線到下列連接埠：

> **Note**
>
> 下列連接埠會根據引擎類型進行分組，因為 Amazon MQ for ActiveMQ 與 Amazon MQ for RabbitMQ 使用不同的連接埠進行連線。

Amazon MQ for ActiveMQ

- Web 主控台 – 連接埠 8162
- OpenWire – 連接埠 61617
- AMQP – 連接埠 5671
- STOMP – 連接埠 61614
- MQTT – 連接埠 8883
- WSS – 連接埠 61619

Amazon MQ for RabbitMQ

- Web 主控台和管理 API – 連接埠 443 和 15671
- AMQP – 連接埠 5671

5. 針對您的代理程式引擎類型執行下列網路連線測試。

> **Note**
>
> 對於沒有公開存取性的代理程式，請從與 Amazon MQ 代理程式相同的 Amazon VPC 內的 Amazon EC2 執行個體執行測試，然後評估回應。

Amazon MQ for ActiveMQ

測試 Amazon MQ for ActiveMQ 代理程式的網路連線能力

1. 開啟新的終端機或命令列視窗。
2. 執行下列 `nslookup` 命令來查詢代理程式 DNS 記錄。對於作用中/待命部署，測試作用中端點和待命端點。作用中/待命端點會以新增至唯一代理程式 ID 的尾碼 `-1` 或 `-2` 識別。以您的資訊取代端點。

```
$ nslookup b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-west-2.amazonaws.com
```

如果查詢成功，您會看到類似以下的輸出。

```
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
```

解析的 IP 地址應符合 Amazon MQ 主控台中提供的 IP 位址。這表示 DNS 伺服器上的網域名稱解析正確，而且您可繼續下一個步驟。

3. 執行下列 `telnet` 命令來測試代理程式的網路路徑。以您的資訊取代端點。以 Web 主控台的連接埠號碼 8162，或其他線路層級連接埠取代*###*，視需要測試其他通訊協定。

> **Note**
>
> 對於作用中/待命部署，如果您使用待命端點執行 `telnet`，則會收到 `Connect failed` 錯誤訊息。這是可預期的，由於待命執行個體本身正在執行，但 ActiveMQ 程序並未執行，且無法存取代理程式的 Amazon EFS 儲存磁碟區。對 `-1` 和 `-2` 端點執行命令，以確保您同時測試作用中執行個體和待命執行個體。

```
$ telnet b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-
west-2.amazonaws.com port
```

對於作用中執行個體，您會看到類似以下的輸出。

```
Connected to b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-
west-2.amazonaws.com.
Escape character is '^]'.
```

4. 執行下列其中一項操作。

   - 如果 `telnet` 命令成功，請檢查 `EstablishedConnectionsCount` 指標並確認代理程式尚未達到最大的線路層級連線限制。您也可藉由檢閱代理程式 `General` 日誌，確認是

否已達到限制。如果此指標大於零，則至少有一個用戶端目前連線至代理程式。如果指標顯示零連線，則再次執行 `telnet` 路徑測試，並等待至少一分鐘再中斷連線，因為代理程式指標會每分鐘發佈一次。

- 如果 `telnet` 命令失敗，請檢查代理程式的彈性網路界面狀態，並確認狀態為 `in-use`。針對每個執行個體的網路界面建立 Amazon VPC 流程日誌，並檢閱所產生的流程日誌。尋找在您執行 `telnet` 命令時代理程式的 IP 地址，並確認連線封包為 ACCEPTED，包括傳回封包。如需詳細資訊，以及查看流程日誌範例，請參閱《Amazon VPC 開發人員指南》中的流程日誌記錄範例。

5. 執行下列 `curl` 命令來檢查與 ActiveMQ 管理 Web 主控台的連線。

```
$ curl https://b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-
west-2.amazonaws.com:8162/index.html
```

如果此命令成功，輸出應該是類似以下的 HTML 文件。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://
www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;
 charset=iso-8859-1" />
        <title>Apache ActiveMQ</title>
        ...
```

## Amazon MQ for RabbitMQ

### 測試 Amazon MQ for RabbitMQ 代理程式的網路連線能力

1. 開啟新的終端機或命令列視窗。
2. 執行下列 `nslookup` 命令來查詢代理程式 DNS 記錄。以您的資訊取代端點。

```
$ nslookup b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-west-2.amazonaws.com
```

如果查詢成功，您會看到類似以下的輸出。

```
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. 執行下列 `telnet` 命令來測試代理程式的網路路徑。以您的資訊取代端點。您可以 Web 主控台的連接埠 443 及 5671 取代*###*，以測試線路層級 AMQP 連線。

```
$ telnet b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-
west-2.amazonaws.com port
```

如果命令成功，您會看到類似以下的輸出。

```
Connected to b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-
west-2.amazonaws.com.
Escape character is '^]'.
```

> **Note**
>
> Telnet 連線會在幾秒鐘後自動關閉。

4. 執行下列其中一項操作。

- 如果 `telnet` 命令成功，請檢查 `ConnectionCount` 指標，並確認代理程式尚未達到 `max-connections` 預設政策中設定的值。您也可藉由檢閱代理程式 `Connection.log` 日誌群組，確認是否已達到限制。如果此指標大於零，則至少有一個用戶端目前連線至代理程式。如果指標顯示零連線，則再次執行 `telnet` 路徑測試。如果連線在代理程式將新的連線指標發佈至 CloudWatch 之前關閉，您可能需要重複此程序。指標每分鐘發佈一次。
- 對於沒有公開存取性的代理程式，如果 `telnet` 命令失敗，請檢查代理程式的彈性網路界面狀態，並確認狀態為 `in-use`。針對每個網路界面建立 Amazon VPC 流程日誌，並檢閱所產生的流程日誌。尋找在叫用 `telnet` 命令時代理程式的私有 IP 地址，並確認連線封包為 `ACCEPTED`，包括傳回封包。如需詳細資訊，以及查看流程日誌範例，請參閱《Amazon VPC 開發人員指南》中的流程日誌記錄範例。

> **Note**
>
> 此步驟不適用於具有公開存取性的 Amazon MQ for RabbitMQ 代理程式。

5. 執行下列 `curl` 命令來檢查與 RabbitMQ 管理 Web 主控台的連線。

```
$ curl https://b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-
west-2.amazonaws.com:443/index.html
```

如果此命令成功，輸出應該是類似以下的 HTML 文件。

```
<!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>
        ...
```

## 我的代理程式正在執行，我可以使用 **telnet** 驗證連線能力，但我的用戶端無法連線並傳回 SSL 例外狀況。

您的代理程式端點憑證可能已經在代理程式維護時段更新。Amazon MQ 代理程式憑證會定期輪換，以確保代理程式的持續可用性和安全性。

建議使用 Amazon Trust Services 中的 Amazon 根憑證授權機構 (CA)，在用戶端的信任存放區中進行身分驗證。所有 Amazon MQ 代理程式憑證都使用此根 CA 進行簽署。透過使用 Amazon 根 CA，您不再需要在每次代理程式上有憑證更新時下載新的 Amazon MQ 代理程式憑證。

## 我建立了代理程式，但代理程式建立失敗。

如果您的代理程式處於 `CREATION_FAILED` 狀態，請執行下列動作。

- 檢查您的 IAM 許可。若要建立代理程式，必須使用 AWS 管理的 IAM 政策 `AmazonMQFullAccess` 或在您的自訂 IAM 政策中具有正確的 Amazon EC2 許可集。若要進一步了解您所需的 Amazon EC2 許可，請參閱建立 Amazon MQ 代理程式所需的 IAM 許可。

- 檢查您為代理程式選擇的子網路是否位於共用的 Amazon Virtual Private Cloud (VPC) 中。若要在共用的 Amazon VPC 中建立 Amazon MQ 代理程式，您必須在擁有 Amazon VPC 的帳戶中建立它。

## 我的代理程式重新啟動，但我不確定原因。

如果您的代理程式已自動重新啟動，可能是由於以下原因之一。

- 您的代理程式可能因為排定的每週維護時段而重新啟動。Amazon MQ 會定期對訊息代理程式的硬體、作業系統或引擎軟體執行維護。維護的持續時間會有所不同，但最多可能持續兩小時，視您針對訊息代理程式排程的操作而定。代理程式可能會在兩小時維護期間的任何時候重新啟動。如需代理程式維護時段的詳細資訊，請參閱 the section called “維護代理程式”。
- 您的代理程式執行個體類型可能不適合您的應用程式工作負載。例如，在 `mq.t2.micro` 上執行生產工作負載可能會導致代理程式耗盡資源。高 CPU 使用率或高代理程式記憶體使用量可能會導致代理程式意外重新啟動。若要查看代理程式正在使用多少 CPU 和記憶體，請針對您的引擎類型使用下列 CloudWatch 指標。
  - Amazon MQ for ActiveMQ – 檢查 `CpuUtilization`，了解代理程式目前使用的已配置 Amazon EC2 運算單位的百分比。檢查 `HeapUsage`，了解代理程式目前使用的 ActiveMQ JVM 記憶體限制的百分比。
  - Amazon MQ for RabbitMQ – 檢查 `SystemCpuUtilization`，了解代理程式目前使用的已配置 Amazon EC2 運算單位的百分比。檢查 `RabbitMQMemUsed`，了解已使用的 RAM 數量 (以位元組為單位)，並除以 `RabbitMQMemLimit` 來取得 RabbitMQ 節點使用的記憶體百分比。

  如需代理程式執行個體類型以及如何針對工作負載選擇正確執行個體類型的詳細資訊，請參閱 Broker instance types。

# 故障診斷：Amazon MQ for ActiveMQ

請使用此節中的資訊，協助診斷及解決在使用 Amazon MQ for ActiveMQ 代理程式時可能遇到的常見問題。

內容

- 我在執行操作時，在 ActiveMQ 主控台上看到例外狀況 org.apache.jasper.JasperException: An exception occurred processing JSP page。

## 即使我已啟用記錄功能，我也無法在 CloudWatch Logs 中看到代理程式的一般或稽核日誌。

如果您無法在 CloudWatch Logs 中檢視代理程式的日誌，請執行下列動作。

1. 檢查建立或重新啟動代理程式的使用者是否具有 `logs:CreateLogGroup` 許可。在使用者建立或重新啟動代理程式之前，如果您未將 `CreateLogGroup` 許可新增至使用者，則 Amazon MQ 不會建立日誌群組。
2. 檢查您是否已設定以資源為基礎的政策，以允許 Amazon MQ 將日誌發佈到 CloudWatch Logs。若要允許 Amazon MQ 將日誌發佈到 CloudWatch Logs 日誌群組，請設定資源型政策，提供 Amazon MQ 存取以下 CloudWatch Logs API 動作的權限：
   - `CreateLogStream` – 為指定的日誌群組建立 CloudWatch Logs 日誌串流。
   - `PutLogEvents` – 將事件傳送到指定的 CloudWatch Logs 日誌串流。

如需有關設定 Amazon MQ for ActiveMQ 以將日誌發佈至 CloudWatch Logs 的詳細資訊，請參閱設定記錄。

## 代理程式重新啟動或維護時段之後，即使狀態為 **RUNNING**，我仍然無法連線到代理程式。為什麼？

在您初始化代理程式重新啟動後、排定的維護時段完成後、或者在啟動待命執行個體的失敗事件中，您可能會遇到連線問題。在任何一種情況下，代理程式重新啟動後的連線問題很可能是因為代理程式的 Amazon EFS 或 Amazon EBS 儲存磁碟區中存在異常大量的訊息所造成。在重新啟動期間，Amazon MQ 會將持續性訊息從儲存區移至代理程式記憶體。若要確認此診斷，可以在 CloudWatch 上監控您的 Amazon MQ for ActiveMQ 代理程式的下列指標：

- **`StoragePercentUsage`** — 大百分比或接近 100% 可能會導致代理程式拒絕連線。
- **`JournalFilesForFullRecovery`** - 指出在非正常關機並重新啟動後將重播的日誌檔案數量。增加或持續高於 1 的值表示未解決的交易，可能會在重新啟動後造成連線問題。
- **`OpenTransactionCount`** - 重新啟動後的數字大於零，表示代理程式將嘗試存放先前使用的訊息，從而導致連線問題。

若要解決此問題，我們建議您使用 `rollback()` 或 `commit()`，解決 XA 交易。如需詳細資訊並查看使用 `rollback()` 解決 XA 交易的程式碼範例，請參閱復原 XA 交易。

## 我看到我的一些用戶端連線到代理程式，而其他用戶端無法連線。

如果您的代理程式處於 `RUNNING` 狀態，而有些用戶端可以成功連線到代理程式，而有些用戶端則無法連線，您可能已經達到代理程式的線路層級連線限制。若要確認您已達到線路層級連線限制，請執行下列動作：

- 請在 CloudWatch Logs 中檢查 Amazon MQ for ActiveMQ 代理程式的一般代理程式日誌。如果已達到限制，您將在代理程式日誌中看到 `Reached Maximum Connections`。如需 Amazon MQ for ActiveMQ 代理程式的 CloudWatch Logs 的詳細資訊，請參閱the section called “了解 CloudWatch Logs 中的記錄結構”。

達到線路層級連線限制後，代理程式將主動拒絕額外的連入連線。若要解決此問題，我們建議升級代理程式執行個體類型。如需有關選擇工作負載之最佳執行個體類型的詳細資訊，請參閱Broker instance types。

如果您已確認線路層級連線數目小於代理程式連線限制，則問題可能與重新啟動用戶端有關。檢查您的代理程式日誌是否有大量和頻繁的 `... Inactive for longer than 600000 ms - removing ...` 條目。日誌項目表示重新啟動用戶端或連線問題。當用戶端透過 Network Load Balancer (NLB) 與經常中斷連線並重新連線至代理程式的用戶端連線至代理程式時，此效果會更明顯。這在以容器為基礎的用戶端中更為常見。

檢查您的用戶端日誌以取得進一步的詳細資訊。代理程式將在 600000 毫秒後清理非活動的 TCP 連接，並釋放連線通訊端。

## 我在執行操作時，在 ActiveMQ 主控台上看到例外狀況 **`org.apache.jasper.JasperException: An exception occurred processing JSP page`**。

如果您正在使用簡單身分驗證並設定佇列的 `AuthorizationPlugin` 和主題授權，請務必在 XML 組態檔案中使用 `AuthorizationEntries` 元素，並對所有佇列和主題允許 `activemq-webconsole` 群組許可。這可確保 ActiveMQ Web 主控台可以與 ActiveMQ 代理程式進行通訊。

以下 `AuthorizationEntry` 範例會將所有佇列和主題的讀取和寫入許可授予給 `activemq-webconsole` 群組。

```
<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>
```

同樣地，當您的代理程式與 LDAP 整合時，請確保為 amazonmq-console-admins 群組授予許可。如需有關 LDAP 整合的詳細資訊，請參閱 the section called “LDAP 整合的運作方式”。

# 故障診斷：Amazon MQ for RabbitMQ

請使用此節中的資訊，協助診斷及解決在使用 Amazon MQ for RabbitMQ 代理程式時可能遇到的常見問題。

內容

- 我在 CloudWatch 中看不到佇列或虛擬主機的指標。
- 如何在 Amazon MQ for RabbitMQ 中啟用外掛程式？
- 我無法變更代理程式的 Amazon VPC 組態。

## 我在 CloudWatch 中看不到佇列或虛擬主機的指標。

如果您無法在 CloudWatch 中檢視佇列或虛擬主機的指標，請檢查佇列或虛擬主機名稱是否包含任何空格、定位字元或其他非 ASCII 字元。

Amazon MQ 無法會為名稱包含空格、定位字元或其他非 ASCII 字元的虛擬主機和佇列發佈指標。

如需維度名稱的詳細資訊，請參閱《Amazon CloudWatch API 參考》中的維度。

## 如何在 Amazon MQ for RabbitMQ 中啟用外掛程式？

Amazon MQ for RabbitMQ 目前只支援 RabbitMQ 管理、Shovel、聯合、一致雜湊交換外掛程式，這些外掛程式預設啟用。如需有關使用支援的外掛程式的詳細資訊，請參閱 the section called “外掛程式”。

## 我無法變更代理程式的 Amazon VPC 組態。

建立代理程式後，Amazon MQ 不支援變更 Amazon VPC 組態。請注意，您需要使用新的 Amazon VPC 組態建立新代理程式，並使用新代理程式連線 URL 更新用戶端連線 URL。

# 故障診斷：需執行的 Amazon MQ 動作的代碼

如果您的代理程式處於運作不佳狀態並需要一組動作來恢復正常狀態，則 Amazon MQ 會傳回某些 API 操作的例外狀況，例如 `RebootBroker`。例外狀況包含特定需執行的動作代碼，可協助您識別根本原因並解決問題，以復原您的代理程式。

使用以下主題清單來識別您收到的需執行的動作代碼，並進一步了解我們為解決您的問題所建議的步驟。

需執行的動作代碼

- Amazon MQ for RabbitMQ：高記憶體警示
- Amazon MQ for RabbitMQ：無效的 AWS Key Management Service 金鑰
- Amazon MQ for ActiveMQ：已刪除彈性網路介面警示
- Amazon MQ for ActiveMQ：代理程式記憶體不足警示
- Amazon MQ for RabbitMQ：磁碟限制警示

## Amazon MQ for RabbitMQ：高記憶體警示

當代理程式的記憶體用量 (由 CloudWatch 指標 `RabbitMQMemUsed` 識別) 超過記憶體上限 (由 `RabbitMQMemLimit` 識別) 時，RabbitMQ 會發出高記憶體警示。`RabbitMQMemLimit` 由 Amazon MQ 設定，並根據每個主機執行個體類型的可用記憶體進行了專門調校。

發出高記憶體警示的 Amazon MQ for RabbitMQ 代理程式將阻止所有正在發佈訊息的用戶端。由於記憶體用量較高，您的代理程式還可能遇到其他問題，這些問題會讓警示的診斷和解決複雜化。

由於記憶體用量較高而無法完成啟動的單一執行個體代理程式可能會進入重新啟動迴圈，在此期間，與代理程式的交互會受到限制。在叢集部署中，佇列可能會遇到不同節點上之複本之間的訊息同步暫停。暫停的佇列同步可防止佇列訊息取用，必須在解決記憶體警示時單獨進行解決。

Amazon MQ 不會重新啟動遇到高記憶體警示的代理程式，並且會傳回 `RebootBroker` API 操作的例外狀況 (只要代理程式繼續發出警示)。

使用本區段中的資訊可協助您診斷和解決代理程式發出的 RabbitMQ 高記憶體警示。

> **Note**
>
> 採取所需的動作之後，RABBITMQ_MEMORY_ALARM 狀態最多可能需要數小時才會解除。

> **Note**
>
> 您不能將代理程式從 `mq.m5.` 執行個體類型降級至 `mq.t3.micro` 執行個體類型。如果您希望降級，則必須刪除您的代理程式，並建立一個新的。

主題



## 使用 RabbitMQ Web 主控台診斷高記憶體警示

RabbitMQ Web 主控台可以產生並顯示每個節點的詳細記憶體用量資訊。您可以透過以下操作找到此資訊：

1. 登入 AWS Management Console 並打開代理程式的 RabbitMQ Web 主控台。
2. 在 RabbitMQ 主控台上，在 Overview (概觀) 頁面上，從 Nodes (節點) 清單中選擇節點名稱。
3. 在節點詳細資訊頁面上，選擇 Memory details (記憶體詳細資訊) 展開區段以檢視節點的記憶體用量資訊。

RabbitMQ 在 Web 主控台中提供的記憶體用量資訊，可協助您確定哪些資源可能會取用過多記憶體並導致高記憶體警示。如需 RabbitMQ Web 主控台可用記憶體用量的詳細資訊，請在 RabbitMQ 伺服器文件網站上參閱有關記憶體使用的推理。

## 使用 Amazon MQ 指標診斷高記憶體警示

預設情況下，Amazon MQ 為您的代理程式啟用指標。您可以透過存取 CloudWatch 主控台，或使用 CloudWatch API，檢視您的代理程式指標。以下指標在診斷 RabbitMQ 高記憶體警示時非常實用。

| Amazon MQ CloudWatch 指標 | 記憶體使用較高的原因 | |
|---|---|---|
| MessageCount | 訊息一直存放在記憶體中，直到將其取用或捨棄。較高的訊息計數可能表示資源過度使用，並可能導致高記憶體警示。 | |
| QueueCount | 佇列存放在記憶體中，並且大量佇列可能會導致高記憶體警示。 | |
| ConnectionCount | 用戶端連線使用記憶體，並且過多的同時連線可能會導致高記憶體警示。 | |
| ChannelCount | 與連線類似，使用每個連線建立的通道也存放在節點記憶體中，並且大量的通道可能導致高記憶體警示。 | |
| ConsumerCount | 對於連線至代理程式的每個取用者，在傳遞給取用者之前，將一組數量的訊息從儲存器載入至記憶體中。大量取用者連線可能會導致高記憶體用量，並導致高記憶體警示。 | |
| PublishRate | 發佈訊息會使用代理程式的記憶體。如果向代理程式發佈訊息的速率太高，並且顯著超過了代理程式向取用者傳遞訊息的速率，則代理程式可能會發出高記憶體警示。 | |

## 解決高記憶體警示

對於您識別的每個參與者，我們建議採取以下一組動作來緩解和解決代理程式的高記憶體警示。

| 記憶體使用較高的原因 | Amazon MQ 推薦 | |
|---|---|---|
| 佇列中的訊息數量過多。 | 執行下列任何一項：<br>• 取用已發佈到佇列的訊息。<br>• 從佇列清除訊息。<br>• 從您的代理程式中刪除佇列。 | |
| 代理程式上設定的佇列數量過多。 | 減少佇列數量。 | |
| 建立於代理程式上的連線數量過多。 | 減少連線數量。如需更多詳細資訊，請參閱 the section called “減少連線和通道的數量”。 | |
| 建立於代理程式上的通道數量過多。 | 減少通道數量。如需詳細資訊，請參閱the section called “減少連線和通道的數量”。 | |
| 連線至代理程式的取用者數量過多。 | 減少連線至代理程式的取用者數量。 | |
| 訊息發佈率過高。 | 降低發佈者傳送訊息至代理程式的速率。 | |
| 用戶端連線嘗試率過高。 | 降低用戶端嘗試連線至代理程式以發佈或取用訊息或設定代理程式的頻率。 | |

### 減少連線和通道的數量

您可以透過用戶端應用程式或透過使用 RabbitMQ Web 主控台手動關閉其來關閉您的 Amazon MQ for RabbitMQ 代理程式連線。若要關閉連線，請使用 RabbitMQ Web 主控台執行以下操作。

1. 登入 AWS Management Console 並打開代理程式的 RabbitMQ Web 主控台。
2. 在 RabbitMQ 主控台上，選擇 Connections (連線) 索引標籤。
3. 在 Connections (連線) 頁面上，All connections (所有連線) 下，從清單中選擇您要關閉的連線名稱。
4. 在連線詳細資訊頁面上，選擇 Close this connection (關閉此連線) 以展開區段，然後選擇 Force Close (強制關閉)。或者，您也可以使用自己的描述取代 Reason (原因) 的預設文字。關閉連線時，Amazon MQ for RabbitMQ 會將您指定的原因傳回用戶端。
5. 在對話方塊上選擇 OK (確定) 以確認並關閉連線。

關閉連線時，還會關閉與關閉連線關聯的所有通道。

> **Note**
>
> 您的用戶端應用程式可以設定為在關閉後自動重新建立與代理程式的連線。在這種情況下，從代理程式 Web 主控台關閉連線不足以減少連線或通道計數。

對於沒有公開存取的代理程式，您可以透過拒絕相應訊息通訊協定連接埠 (例如，適用於 AMQP 連線的連接埠 5671) 上的傳入流量來臨時阻止連線。您可以在建立代理程式時阻止您提供給 Amazon MQ 之安全群組中的連接埠。如需有關修改安全群組的詳細資訊，請參閱《Amazon VPC 使用者指南》中的新增規則至安全群組。

### 解決叢集部署中暫停的佇列同步

在解決 RabbitMQ 的高記憶體警示時，您可能會發現無法取用一個或多個佇列上的訊息。這些佇列可能正在同步節點之間的訊息，在此期間，相應的佇列變得不可用於發佈和取用。佇列同步可能由於高記憶體警示而暫停，甚至會導致記憶體警報。

如需停用和重試已暫停佇列同步的相關資訊，請參閱 the section called “解決暫停的佇列同步”。

### 解決單一執行個體代理程式中的重新啟動迴圈

如果重新啟動並且沒有足夠的記憶體啟動，則發出高記憶體警示的 Amazon MQ for RabbitMQ 單一執行個體代理程式有可能變得不可用。這可能導致 RabbitMQ 進入重新啟動迴圈，並阻止與代理程式進

一步交互，直到問題得到解決。如果您的代理程式處於重新啟動迴圈中，則您將無法套用本節之前描述的 Amazon MQ 推薦動作來解決高記憶體警示。

要復原您的代理程式，我們建議升級到具有更大記憶體的較大執行個體類型。與叢集部署不同，您可以在其遇到高記憶體警示時升級單一執行個體代理程式，因為在重新啟動期間節點之間沒有要執行的佇列同步。

## 防止高記憶體警示

對於您識別的每個因素，我們建議採取以下一組動作，以防止和減少 RabbitMQ 高記憶體警示的發生。

| 記憶體使用較高的原因 | Amazon MQ 推薦 |
| --- | --- |
| 佇列中的訊息數量過多。 | 請執行下列操作：<br>• 啟用延遲佇列。<br>• 設定，或降低佇列深度限制。 |
| 代理程式上設定的佇列數量過多。 | 設定，或降低佇列計數限制。 |
| 建立於代理程式上的連線數量過多。 | 設定，或降低連線計數限制。 |
| 建立於代理程式上的通道數量過多。 | 在用戶端應用程式上設定每個連線的最大通道數量。 |
| 連線至代理程式的取用者數量過多。 | 設定小型取用者預擷取限制。 |
| 用戶端連線嘗試率過高。 | 使用壽命較長的連線可減少連線嘗試次數和頻率。 |

解決代理程式的記憶體警示後，您可以將您的主機執行個體類型升級到具有其他資源的執行個體。如需如何更新代理程式執行個體類型的詳細資訊，請參閱 Amazon MQ REST API 參考中的 `UpdateBrokerInput`。

如需代理程式執行個體類型的完整清單，請參閱 the section called “Amazon MQ for RabbitMQ 執行個體類型”。

## Amazon MQ for RabbitMQ：無效的 AWS Key Management Service 金鑰

當使用客戶受管 AWS KMS key (CMK) 建立的代理程式偵測到 AWS Key Management Service (KMS) 金鑰已停用時，Amazon MQ for RabbitMQ 會引發 INVALID_KMS_KEY 關鍵動作所需程式碼。具有 CMK 的 RabbitMQ 代理程式會定期驗證 KMS 金鑰是否已啟用，且代理程式具有所有必要的授權。如果 RabbitMQ 無法驗證金鑰是否已啟用，則代理程式會被隔離，而 RabbitMQ 會傳回 INVALID_KMS_KEY。

如果沒有作用中的 KMS 金鑰，代理程式就沒有客戶受管 KMS 金鑰的基本許可。在您重新啟用金鑰且代理程式重新啟動之前，代理程式無法使用您的金鑰執行加密作業。系統會隔離具有已停用 KMS 金鑰的 RabbitMQ 代理程式，以防止惡化。RabbitMQ 判斷 KMS 金鑰再次處於作用中狀態之後，您的代理程式就會從隔離區中移除。Amazon MQ 不會重新啟動具有已停用 KMS 金鑰的代理程式，而且只要代理程式持續擁有無效的 KMS 金鑰，就會傳回 `RebootBroker` API 操作的例外狀況。

### 診斷和解決 INVALID_KMS_KEY

若要診斷和解決 INVALID_KMS_KEY 動作所需的程式碼，您必須使用 AWS 命令列介面 (CLI) 和 AWS Key Management Service 主控台。

若要重新啟用您的 KMS 金鑰

1. 呼叫 `DescribeBroker` 方法以擷取您 CMK 代理程式的 `kmsKeyId`。
2. 登入 AWS Key Management Service主控台。
3. 在 客戶受管金鑰 頁面上，找出有問題的代理程式的 KMS 金鑰 ID，並確認狀態為 已啟用。
4. 如果您的 KMS 金鑰已停用，請選擇 金鑰動作，然後選擇 啟用，以重新啟用金鑰。重新啟用金鑰後，您必須等待 RabbitMQ 從隔離區中刪除代理程序。

若要驗證必要的授權仍與代理程式的 KMS 金鑰相關聯，請呼叫 `ListGrant` ListGrant 方法以驗證 `mq_rabbit_grant` 和 `mq_grant` 均存在。如果 KMS 授權或金鑰已刪除，則必須刪除該代理程式，並建立一個具有一切必要授權的新代理程式。如需刪除代理程式的步驟，請參閱刪除代理程式。

若要避免 INVALID_KMS_KEY 需要關鍵動作程式碼，請勿手動刪除或停用 KMS 金鑰或 CMK 授權。如果您想要刪除金鑰，請先刪除代理程式。

## Amazon MQ for ActiveMQ：已刪除彈性網路介面警示

當您刪除代理程式的彈性網路介面 (ENI) 時，Amazon MQ for ActiveMQ 會引發 BROKER_ENI_DELETED 警示。當您第一次建立 Amazon MQ 代理程式時，Amazon MQ 會在 Virtual Private Cloud (VPC)中您的帳戶之下佈建彈性網路界面，因此，需要一些 EC2 許可。

您不得修改或刪除這個網路界面。修改或刪除網路界面可能導致永久遺失 VPC 與代理程式之間的連線。如果您想要刪除網路介面，您必須先刪除代理程式。

## Amazon MQ for ActiveMQ：代理程式記憶體不足警示

當代理程式由於記憶體容量不足而歷經重新啟動時，Amazon MQ for ActiveMQ 將發出 BROKER_OOM 警示。當代理程式處於重新啟動迴圈 (也稱為反彈迴圈) 時，代理程式會在短時間範圍內啟動重複的復原嘗試。由於記憶體容量不足而無法完成啟動的代理程式可以進入重新啟動迴圈，在此期間與代理程式的互動受到限制。

預設情況下，Amazon MQ 為您的代理程式啟用指標。您可以透過存取 Amazon CloudWatch 主控台，或使用 CloudWatch API，檢視您的代理程式指標。以下指標在診斷 ActiveMQ BROKER_OOM 警示時非常實用：

| Amazon MQ CloudWatch 指標 | 記憶體使用較高的原因 |
| --- | --- |
| `TotalMessageCount` | 訊息一直存放在記憶體中，直到將其取用或捨棄。較高的訊息計數可能表示資源過度使用，並可能導致高記憶體警示。 |
| `HeapUsage` | 代理程式目前使用 ActiveMQ JVM 記憶體限制。較高的百分比表示代理程式正在使用大量資源，並且可能導致 OOM 警示。 |
| `ConnectionCount` | 用戶端連線使用記憶體，並且過多的同時連線可能會導致高記憶體警示。 |

| Amazon MQ CloudWatch 指標 | 記憶體使用較高的原因 | |
|---|---|---|
| CpuUtilization | 代理當前正在使用的已分配 EC2 運算單位的百分比。 | |
| TotalConsumerCount | 對於連線至代理程式的每個取用者，在傳遞給取用者之前，將一組數量的訊息從儲存器載入至記憶體中。大量取用者連線可能會導致高記憶體用量，並導致高記憶體警示。 | |

為了防止重新啟動迴圈並避免 BROKER_OOM 警示，請確保訊息快速消耗。您可以選擇最有效的代理程式執行個體類型，並清除無效字母佇列以捨棄無法傳遞或過期的訊息來執行此操作。您可以在 Amazon MQ for ActiveMQ 最佳實務進一步了解如何確保有效的效能。

## Amazon MQ for RabbitMQ：磁碟限制警示

磁碟限制警示表示 RabbitMQ 節點使用的磁碟容量已減少，這是因為新增訊息時未消耗大量訊息。當代理程式的可用磁碟空間 (由 Amazon CloudWatch 指標 RabbitMQDiskFree 識別) 達到磁碟限制 (由 RabbitMQDiskFreeLimit 識別) 時，RabbitMQ 將發出磁碟限制警示。RabbitMQDiskFreeLimit 由 Amazon MQ 設定，並已根據每個代理程式執行個體類型的可用磁碟空間進行定義.

發出磁碟限制警示的 Amazon MQ for RabbitMQ 代理程式將無法用於發布新訊息。在叢集中執行 RabbitMQ 時，磁碟警示是泛叢集範圍的。如果一個節點低於限制，所有其他節點將阻止傳入訊息。由於磁碟空間不足，您的代理程式還可能遇到其他問題，這些問題會讓警示的診斷和解決複雜化。

Amazon MQ 不會重新啟動遇到磁碟警示的代理程式，並且會傳回 RebootBroker API 操作的例外狀況 (只要代理程式繼續發出警示)。

> **Note**
>
> 您不能將代理程式從 mq.m5 執行個體類型降級至 mq.t3.micro 執行個體類型。如果您希望降級，則必須刪除您的代理程式，並建立一個新的。

## 診斷和定址磁碟限制警示

預設情況下，Amazon MQ 為您的代理程式啟用指標。您可以透過存取 Amazon CloudWatch 主控台，或使用 CloudWatch API，檢視您的代理程式指標。在診斷 RabbitMQ 磁碟限制警示時，`MessageCount` 是一個實用的指標。訊息一直存放在記憶體中，直到將其取用或捨棄。較高的訊息計數表示磁碟儲存體警示的磁碟存放區過度使用，並可能導致磁碟警示。

若要診斷磁碟限制警示，請使用 Amazon MQ 管理主控台執行以下操作：

- 取用已發佈到佇列的訊息。
- 從佇列清除訊息。
- 從您的代理程式中刪除佇列。

> **Note**
>
> 採取所需的動作之後，RABBITMQ_DISK_ALARM 狀態最多可能需要數小時才會解除。

如要避免磁碟限制警示再度發生，您可以將您的主機執行個體類型升級到具有其他資源的執行個體。如需如何更新代理程式執行個體類型的詳細資訊，請參閱 Amazon MQ REST API 參考中的 `UpdateBrokerInput`。

# 相關資源

## Amazon MQ 資源

下表列出與 Amazon MQ 搭配運作的有用資源。

| 資源 | 描述 |
| --- | --- |
| Amazon MQ REST API 參考 | 描述 REST 資源、範例請求、HTTP 方法、結構描述、參數，以及服務傳回的錯誤。 |
| AWS CLI 命令參考中的 Amazon MQ | 描述您可以用來使用訊息代理程式的 AWS CLI 命令。 |
| AWS CloudFormation 使用者指南中的 Amazon MQ | `AWS::Amazon MQ::Broker` 資源可讓您建立 Amazon MQ 代理程式、新增組態變更或修改指定代理程式的使用者、傳回指定代理程式的相關資訊，以及刪除指定的代理程式。<br><br>`AWS::Amazon MQ::Configuration` 資源可讓您建立 Amazon MQ 組態、新增組態變更或修改使用者，以及傳回所指定組態的相關資訊。 |
| 區域與終端節點 | Amazon MQ 區域與端點的相關資訊。 |
| 產品頁面 | Amazon MQ 相關資訊的主要網頁。 |
| 開發論壇 | 以社群為基礎的論壇，供開發人員討論 Amazon MQ 相關技術問題。 |
| AWS Premium Support 資訊 | AWS Premium Support 相關資訊的主要網頁，本支援服務是一對一的快速回應支援管道，可協助您在 AWS 基礎設施服務建立並執行應用程式 |

## Amazon MQ for ActiveMQ 資源

下表列出與 Apache ActiveMQ 搭配使用的有用資源。

| 資源 | 描述 |
| --- | --- |
| Apache ActiveMQ 入門指南 | Apache ActiveMQ 的官方文件。 |
| ActiveMQ in Action | Apache ActiveMQ 的指南，其中涵蓋了 JMS 訊息、連接器、訊息持久性、身分驗證和授權的剖析。 |
| Cross-Language Clients | 列出程式設計語言和對應的 Apache ActiveMQ 程式庫。另請參閱 ActiveMQ Client 和 QpidJMS Client。 |

# Amazon MQ for RabbitMQ 資源

下表列出與 RabbitMQ 搭配運作的有用資源。

| 資源 | 描述 |
| --- | --- |
| RabbitMQ 入門指南 | RabbitMQ 的官方文件。 |
| RabbitMQ 用戶端程式庫和開發人員工具 | 使用各種程式設計語言和平台搭配 RabbitMQ 運作之官方支援的用戶端程式庫和開發者工具指南。 |
| RabbitMQ 最佳實務 | CloudAMQP 搭配 RabbitMQ 運作的最佳實務和建議指南。 |

# Amazon MQ 版本備註

下表列出 Amazon MQ 的功能發佈與改進。如需《Amazon MQ 開發人員指南》的變更內容，請參閱 Amazon MQ 文件歷史記錄。

| 日期 | 文件更新 |
| --- | --- |
| 2023年3月4日 | 適用於兔子 MQ 的 Amazon MQ 現在支持兔子 MQ 3.11.28。<br><br>如需此版本中修正及功能的詳細資訊，請參閱下列項目：<br><br>• RabbitMQ 伺服器儲存庫上的版本說明 GitHub<br>• RabbitMQ 變更記錄<br><br>如需支援的 Amazon MQ for RabbitMQ 版本和代理程式升級詳細資訊，請參閱管理 Amazon MQ for RabbitMQ 引擎版本。 |
| 2024 年 1 月 19 日 | 適用於 RabbitMQ 的 Amazon MQ 不支援使用者名稱「訪客」，而且會在您建立新的代理程式時刪除預設來賓帳戶。Amazon MQ 也會定期刪除任何客戶建立的名為「訪客」的帳戶。 |
| 2023 年 12 月 15 日 | Amazon MQ 現可於以色列 (特拉維夫) 區域取得。如需可用區域的資訊，請參閱《AWS 一般參考指南》中的「AWS 區域和端點」。 |
| 2023 年 12 月 11 日 | Amazon MQ for RabbitMQ 現在支援 RabbitMQ 3.10.25。<br><br>如需此版本中修正及功能的詳細資訊，請參閱下列項目：<br><br>• RabbitMQ 伺服器儲存庫上的版本說明 GitHub<br>• RabbitMQ 變更記錄<br><br>如需支援的 Amazon MQ for RabbitMQ 版本和代理程式升級詳細資訊，請參閱管理 Amazon MQ for RabbitMQ 引擎版本。 |
| 2023 年 10 月 26 日 | Amazon MQ 隨重大更新發行了最新的 ActiveMQ 次要版本 5.15.16、5.16.7、5.17.6。我們已棄用較舊的 ActiveMQ 次要版本，並且會將任何 5.15 版 |

| 日期 | 文件更新 |
| --- | --- |
| | 本上的所有代理程式更新為 5.15.16，或 5.16 更新為 5.16.7，以及 5.17 更新為 5.17.6。<br><br>如需 ActiveMQ 代理程式的詳細資訊，請參閱 管理 Amazon MQ for ActiveMQ 引擎版本。 |
| 2023 年 9 月 27 日 | Amazon MQ for RabbitMQ 現在支援 RabbitMQ 3.11.20。<br><br>如需此版本中修正及功能的詳細資訊，請參閱下列項目：<br><br>• RabbitMQ 伺服器儲存庫上的版本說明 GitHub<br>• RabbitMQ 變更記錄<br><br>如需支援的 Amazon MQ for RabbitMQ 版本和代理程式升級詳細資訊，請參閱 管理 Amazon MQ for RabbitMQ 引擎版本。 |
| 2023 年 7 月 27 日 | Amazon MQ for RabbitMQ 現在支援 RabbitMQ 3.11.16。<br><br>如需此版本中修正及功能的詳細資訊，請參閱下列項目：<br><br>• RabbitMQ 伺服器儲存庫上的版本說明 GitHub<br>• RabbitMQ 變更記錄<br><br>如需支援的 Amazon MQ for RabbitMQ 版本和代理程式升級詳細資訊，請參閱 管理 Amazon MQ for RabbitMQ 引擎版本。 |
| 2023 年 7 月 27 日 | Amazon MQ for RabbitMQ 現在支援建立和套用組態至您的 RabbitMQ 代理程式。<br><br>如需新增組態至代理程式的詳細資訊，請參閱 RabbitMQ Broker Configurations。<br><br>如需有關此功能的詳細資訊，請參閱：<br><br>• 操作員政策<br>• 操作員政策的變更 |

| 日期 | 文件更新 |
| --- | --- |
| 2023 年 6 月 23 日 | Amazon MQ 現在支援 ActiveMQ 5.17.3，這是一個新的次要引擎版本發行。此版本支援 Amazon MQ 的全新跨區域資料複寫 (CRDR) 功能。<br><br>如需詳細資訊，請參閱下列內容：<br><br>• 若要開始使用 CRDR，請參閱《開發人員指南》中的 Amazon MQ for ActiveMQ 跨區域資料複寫。<br>• ActiveMQ 5.17.3 發行頁面<br>• 管理 Amazon MQ for ActiveMQ 引擎版本<br>• 升級 Amazon MQ 代理程式引擎版本<br>• 使用 Spring XML 組態檔案 |
| 2023 年 6 月 21 日 | 適用於 ActiveMQ 的 Amazon MQ 現在提供跨區域資料複寫 (CRDR) 功能，允許從主要區域中的主要代理程式將非同步訊息複寫到複本 AWS 區域中的複本代理程式。如果主要區域中的主要代理程式失敗，您可以藉由啟動切換或容錯移轉，將次要區域中的複本代理程式提升為主要代理程式。<br><br>若要開始使用 CRDR，請參閱《開發人員指南》中的 Amazon MQ for ActiveMQ 跨區域資料複寫。 |
| 2023 年 5 月 18 日 | Amazon MQ 現已在下列區域提供：<br><br>• 亞太區域 (墨爾本)<br>• 亞太區域 (海德拉巴)<br>• 歐洲 (西班牙)<br>• 歐洲 (蘇黎世)<br><br>如需可用區域的資訊，請參閱《AWS 一般參考指南》中的「AWS 區域和端點」。 |

| 日期 | 文件更新 |
| --- | --- |
| 2023 年 4 月 14 日 | Amazon MQ for RabbitMQ 現在支援 RabbitMQ 3.9.27 版。<br><br>如需此版本中修正及功能的詳細資訊，請參閱下列項目：<br><br>• RabbitMQ 伺服器儲存庫上的版本資訊 3.9.27 版本說明 GitHub<br>• RabbitMQ 變更記錄<br><br>如需支援的 Amazon MQ for RabbitMQ 版本和代理程式升級詳細資訊，請參閱管理 Amazon MQ for RabbitMQ 引擎版本。 |
| 2023 年 4 月 14 日 | Amazon MQ for RabbitMQ 現在支援 RabbitMQ 3.10.20 版。<br><br>如需此版本中修正及功能的詳細資訊，請參閱下列項目：<br><br>• RabbitMQ 伺服器儲存庫上的版本說明 GitHub<br>• RabbitMQ 變更記錄<br><br>如需支援的 Amazon MQ for RabbitMQ 版本和代理程式升級詳細資訊，請參閱管理 Amazon MQ for RabbitMQ 引擎版本。 |
| 2023 年 3 月 31 日 | Amazon MQ for RabbitMQ 已停用 RabbitMQ 引擎 3.10.17 版<br><br>Amazon MQ for RabbitMQ 團隊和 RabbitMQ 開放原始碼維護者在 3.10.17 版發現了 RabbitMQ 管理主控台的問題。Amazon MQ 已經撤回了這個版本。為了減輕此問題的影響，請使用版本 3.10.20 建立新的代理程式，同時我們努力支援 RabbitMQ 的新修補程式版本。我們建議您啟用自動次要版本升級選項，以自動取得最新的錯誤修正、安全性更新和效能增強功能。<br><br>如需有關可用 Amazon MQ for RabbitMQ 版本的詳細資訊，請參閱 Amazon MQ for RabbitMQ 引擎版本。 |

| 日期 | 文件更新 |
| --- | --- |
| 2023 年 3 月 1 日 | Amazon MQ for RabbitMQ 現在支援 RabbitMQ 3.10.17 版。<br><br>如需此版本中修正及功能的詳細資訊，請參閱下列項目：<br><br>• RabbitMQ 伺服器儲存庫上的版本說明 GitHub<br>• RabbitMQ 變更記錄<br><br>如需支援的 Amazon MQ for RabbitMQ 版本和代理程式升級詳細資訊，請參閱管理 Amazon MQ for RabbitMQ 引擎版本。 |
| 2023 年 2 月 21 日 | 適用於 RabbitMQ 的 Amazon MQ 現在與 AWS Key Management Service (KMS) 整合以提供伺服器端加密。您現在可以選取自己的客戶管理 CMK，或在您的 AWS KMS 帳戶中使用 AWS 受管 KMS 金鑰。如需詳細資訊，請參閱 靜態加密。<br><br>Amazon MQ 支援以下列方式使用 AWS KMS 金鑰。<br><br>• Amazon MQ owned KMS key (default) (Amazon MQ 擁有的 KMS 金鑰 (預設)) — 此金鑰由 Amazon MQ 擁有及管理，不在您的帳戶中。<br>• AWS 受管 KMS 金鑰 — AWS 受管 KMS 金鑰 (aws/mq) 是您帳戶中的 KMS 金鑰，由 Amazon MQ 代表您建立、管理和使用。<br>• 選取現有客戶管理的 KMS 金鑰 — 客戶管理的 KMS 金鑰由您在 AWS Key Management Service (KMS) 中建立和管理。 |
| 2023 年 1 月 13 日 | Amazon MQ for RabbitMQ 現在支援 RabbitMQ 3.8.34 版。<br><br>如需此版本中修正及功能的詳細資訊，請參閱下列項目：<br><br>• RabbitMQ 伺服器儲存庫上的版本說明 GitHub<br>• RabbitMQ 變更記錄<br><br>如需支援的 Amazon MQ for RabbitMQ 版本和代理程式升級詳細資訊，請參閱管理 Amazon MQ for RabbitMQ 引擎版本。 |

| 日期 | 文件更新 |
| --- | --- |
| 2022 年 12 月 15 日 | Amazon MQ for RabbitMQ 現在支援 RabbitMQ 3.9.24 版。<br><br>如需此版本中修正及功能的詳細資訊，請參閱下列項目：<br><br>• RabbitMQ 伺服器儲存庫上的版本說明 3.9.24 GitHub<br>• RabbitMQ 變更記錄<br><br>如需支援的 Amazon MQ for RabbitMQ 版本和代理程式升級詳細資訊，請參閱管理 Amazon MQ for RabbitMQ 引擎版本。 |
| 2022 年 12 月 13 日 | Amazon MQ 現已在中東 (阿拉伯聯合大公國) 區域提供。如需可用區域的資訊，請參閱《AWS 一般參考指南》中的「AWS 區域和端點」。 |
| 2022 年 11 月 14 日 | Amazon MQ for RabbitMQ 現在支援 3.10，這是一個主要引擎版本發行。您現在可以在 RabbitMQ 佇列上啟用傳統佇列第 2 版 (CQv2)。不支援從 3.8 直接更新到 3.10。如需詳細資訊，請參閱下列內容：<br><br>• RabbitMQ 3.10.10 版本備註<br>• RabbitMQ 變更記錄<br><br>如需支援的 Amazon MQ for RabbitMQ 版本和代理程式升級詳細資訊，請參閱管理 Amazon MQ for RabbitMQ 引擎版本。 |
| 2022 年 11 月 9 日 | Amazon MQ 現在支援 ActiveMQ 5.17.2，這是一個新的次要引擎版本發行。如需詳細資訊，請參閱下列內容：<br><br>• ActiveMQ 5.17.2 發行頁面<br>• 管理 Amazon MQ for ActiveMQ 引擎版本<br>• 升級 Amazon MQ 代理程式引擎版本<br>• 使用 Spring XML 組態檔案 |

| 日期 | 文件更新 |
| --- | --- |
| 2022 年 8 月 17 日 | Amazon MQ 現在支援 ActiveMQ 5.17.1，這是一個新的主要引擎版本發行。如需詳細資訊，請參閱下列內容：<br>• ActiveMQ 5.17.1 發行頁面<br>• 管理 Amazon MQ for ActiveMQ 引擎版本<br>• 升級 Amazon MQ 代理程式引擎版本<br>• 使用 Spring XML 組態檔案 |
| 2022 年 7 月 14 日 | Amazon MQ 現在支援 ActiveMQ 5.16.5，這是一個次要引擎版本發行。如需詳細資訊，請參閱下列內容：<br>• ActiveMQ 5.16.5 發行頁面<br>• 管理 Amazon MQ for ActiveMQ 引擎版本<br>• 使用 Spring XML 組態檔案<br>• 升級 Amazon MQ 代理程式引擎版本 |
| 2022 年 5 月 4 日 | Amazon MQ 為代理程式組態中的 `networkConnector` 元素新增包容性語言<br>• 建立和設定 Amazon MQ 代理程式網路 |
| 2022 年 4 月 25 日 | Amazon MQ 此版本新增 `CRITICAL_ACTION_REQUIRED` 代理程式狀態和 `ActionRequired` API 屬性。`CRITICAL_ACTION_REQUIRED` 會在您的代理程式降級時通知您。`ActionRequired` 為您提供了一個代碼，您可以用它在開發人員指南中尋找有關如何解決問題的說明。<br>• the section called “故障診斷：需執行的 Amazon MQ 動作的代碼”<br>• Amazon MQ API 參考中的`ActionRequired` 文件。 |
| 2022 年 4 月 20 日 | Amazon MQ 現在支援 ActiveMQ 5.16.4，這是一個次要引擎版本發行。如需詳細資訊，請參閱下列內容：<br>• ActiveMQ 5.16.4 發行頁面<br>• 管理 Amazon MQ for ActiveMQ 引擎版本<br>• 使用 Spring XML 組態檔案<br>• 升級 Amazon MQ 代理程式引擎版本 |

| 日期 | 文件更新 |
| --- | --- |
| 2022 年 3 月 1 日 | Amazon MQ 現可於亞太區域 (雅加達) 區域使用。如需可用區域的資訊，請參閱《AWS 一般參考指南》中的「AWS 區域和端點」。 |
| 2022 年 2 月 25 日 | Amazon MQ for RabbitMQ 現在支援 RabbitMQ 3.8.27 版。<br><br>如需此版本中修正及功能的詳細資訊，請參閱下列項目：<br><br>• RabbitMQ 伺服器儲存庫上的版本資訊 3.8.27 版本說明 GitHub<br>• RabbitMQ 變更記錄<br><br>如需支援的 Amazon MQ for RabbitMQ 版本和代理程式升級詳細資訊，請參閱管理 Amazon MQ for RabbitMQ 引擎版本。 |
| 2022 年 2 月 16 日 | Amazon MQ 現可於非洲 (開普敦) 區域使用。如需可用區域的資訊，請參閱《AWS 一般參考指南》中的「AWS 區域和端點」。 |
| 2022 年 2 月 14 日 | Amazon MQ for RabbitMQ 現在支援 RabbitMQ 3.9.13 版。自動次要版本升級不能用於 Rabbit 3.8 至 3.9 的升級作業。若要這麼做，請手動升級您的代理程式。<br><br>如需 RabbitMQ 3.9 中引入的新功能的詳細資訊，請參閱網站上版本 3.9.0 的版本說明頁面。 GitHub<br><br>Note<br>目前，Amazon MQ 不支援串流，也無法在 RabbitMQ 3.9 中推出的 JSON 中使用結構化日誌記錄。<br><br>如需此版本中修正及功能的詳細資訊，請參閱下列項目：<br><br>• 有關 RabbitMQ 伺服器儲存庫的發行說明 GitHub<br>• RabbitMQ 變更記錄<br><br>如需支援的 Amazon MQ for RabbitMQ 版本和代理程式升級詳細資訊，請參閱管理 Amazon MQ for RabbitMQ 引擎版本。 |

| 日期 | 文件更新 |
| --- | --- |
| 2022 年 2 月 7 日 | Amazon MQ for RabbitMQ 推出全新代理程式指標，允許您監控叢集部署中所有三個節點的平均資源利用率。<br><br>如需詳細資訊，請參閱下列內容：<br><br>• the section called “記錄和監控 Amazon MQ for RabbitMQ 代理程式” |
| 2022 年 1 月 18 日 | Amazon MQ for RabbitMQ 現在支援 RabbitMQ 3.8.26 版。<br><br>如需此版本中修正及功能的詳細資訊，請參閱下列項目：<br><br>• RabbitMQ 伺服器儲存庫上的版本資訊 3.8.26 版本說明 GitHub<br>• RabbitMQ 變更記錄<br><br>如需支援的 Amazon MQ for RabbitMQ 版本和代理程式升級詳細資訊，請參閱管理 Amazon MQ for RabbitMQ 引擎版本。 |
| 2022 年 1 月 13 日 | Amazon MQ 推出 RABBITMQ_MEMORY_ALARM 狀態碼，以便在您的代理程式發出高記憶體警示並處於運作不佳狀態時通知您。Amazon MQ 提供了詳細的資訊和建議，以協助您診斷、解決和防止高記憶體警示。如需更多資訊，請參閱下列內容。<br><br>• the section called “ RABBITMQ_MEMORY_ALARM ” |
| 2022 年 1 月 6 日 | 當您設定適用於 ActiveMQ 代理程式的 Amazon MQ CloudWatch 日誌時，Amazon MQ 支援在以 IAM 資源為基礎的政策中使用aws:SourceArn 和aws:SourceAccount 全域條件內容金鑰，以避免混淆的副問題。如需更多資訊，請參閱下列內容。<br><br>• the section called “預防跨服務混淆代理人” |
| 2021 年 12 月 20 日 | Amazon MQ for ActiveMQ 推出了一組新的指標，允許您監控使用不同受支援傳輸通訊協定與代理程式建立的最大連線數量，以及額外的新指標，允許您監控在代理程式網路中與代理程式連線的節點數。如需更多資訊，請參閱下列內容。<br><br>• the section called “記錄和監控 Amazon MQ for ActiveMQ 代理程式” |

| 日期 | 文件更新 |
| --- | --- |
| 2021 年 11 月 16 日 | Amazon MQ for RabbitMQ 現在支援 RabbitMQ 3.8.23 版。<br><br>如需此版本中修正及功能的詳細資訊，請參閱下列項目：<br><br>• RabbitMQ 伺服器儲存庫上的版本說明 GitHub<br>• RabbitMQ 變更記錄<br><br>如需支援的 Amazon MQ for RabbitMQ 版本和代理程式升級詳細資訊，請參閱管理 Amazon MQ for RabbitMQ 引擎版本。 |
| 2021 年 10 月 12 日 | Amazon MQ 現在支援 ActiveMQ 5.16.3，這是一個次要引擎版本發行。如需詳細資訊，請參閱下列內容：<br><br>• ActiveMQ 5.16.3 發行頁面<br>• 管理 Amazon MQ for ActiveMQ 引擎版本<br>• 升級 Amazon MQ 代理程式引擎版本<br>• 使用 Spring XML 組態檔案 |
| 2021 年 9 月 8 日 | Amazon MQ for RabbitMQ 現在支援 RabbitMQ 3.8.22 版。<br><br>此版本包含的修正適用於使用每個訊息 TTL (存留時間) 的佇列相關問題，該問題是在先前支援的 RabbitMQ 3.8.17 版中發現。我們建議您現有的代理程式升級至 3.8.22 版。<br><br>如需此版本中修正及功能的詳細資訊，請參閱下列項目：<br><br>• RabbitMQ 伺服器儲存庫上的版本說明 GitHub<br>• RabbitMQ 變更記錄<br><br>如需支援的 Amazon MQ for RabbitMQ 版本和代理程式升級詳細資訊，請參閱管理 Amazon MQ for RabbitMQ 引擎版本 |
| 2021 年 8 月 25 日 | 適用於 RabbitMQ 的 Amazon MQ 已暫時停用 RabbitMQ 引擎 3.8.17 版，這是因為使用每個訊息 (TTL) 的佇列發現問題。 time-to-live 我們建議使用 3.8.11 版。 |

| 日期 | 文件更新 |
| --- | --- |
| 2021 年 7 月 29 日 | Amazon MQ for RabbitMQ 現在支援 RabbitMQ 3.8.17 版。如需此更新中包含的修正和功能詳細資訊，請參閱下列項目：<br>• RabbitMQ 伺服器儲存庫上的版本說明 GitHub<br>• RabbitMQ 變更記錄<br>• 管理 Amazon MQ for RabbitMQ 引擎版本 |
| 2021 年 7 月 16 日 | 您現在可以使用 AWS Management Console、 AWS CLI或 Amazon MQ API 調整 Amazon MQ 代理程式的維護時段。若要進一步了解代理程式維護時段，請參閱下列項目。<br>• 維護 Amazon MQ 代理程式 |
| 2021 年 7 月 6 日 | Amazon MQ for RabbitMQ 引入對一致雜湊交換類型的支援。一致雜湊交換會根據從訊息路由索引鍵計算的雜湊值，將訊息路由傳送到佇列。如需詳細資訊，請參閱下列內容：<br>• 一致雜湊交換外掛程式<br>• RabbitMQ 儲存庫上的一致雜湊交換類型 GitHub |
| 2021 年 6 月 7 日 | Amazon MQ 現在支援 ActiveMQ 5.16.2，這是一個新的主要引擎版本發行。如需詳細資訊，請參閱下列內容：<br>• ActiveMQ 5.16.2 發行頁面<br>• 管理 Amazon MQ for ActiveMQ 引擎版本<br>• 升級 Amazon MQ 代理程式引擎版本<br>• 使用 Spring XML 組態檔案 |
| 2021 年 5 月 26 日 | Amazon MQ for RabbitMQ 現已在中國 (北京) 和中國 (寧夏) 區域提供。如需可用區域的相關資訊，請參閱 AWS 區域與端點。 |

| 日期 | 文件更新 |
| --- | --- |
| 2021 年 5 月 18 日 | Amazon MQ for RabbitMQ 會實作代理程式預設值。<br><br>當您第一次建立代理程式時，Amazon MQ 會根據您選擇的執行個體類型和部署模式建立一組代理程式政策和虛擬主機限制，以便將代理程式的效能最佳化。如需詳細資訊，請參閱下列內容：<br><br>• Amazon MQ for RabbitMQ 代理程式預設值 |
| 2021 年 5 月 5 日 | Amazon MQ 現在支援 ActiveMQ 5.15.15。如需詳細資訊，請參閱下列內容：<br><br>• ActiveMQ 5.15.15 發行頁面<br>• 管理 Amazon MQ for ActiveMQ 引擎版本<br>• 使用 Spring XML 組態檔案 |
| 2021 年 5 月 5 日 | Amazon MQ 開始追蹤 AWS 受管政策的變更。如需詳細資訊，請參閱下列內容：<br><br>• the section called “AWS 受管政策” |
| 2021 年 4 月 14 日 | Amazon MQ 現已在中國 (北京) 和中國 (寧夏) 區域提供。如需可用區域的相關資訊，請參閱 AWS 區域與端點。 |
| 2021 年 4 月 7 日 | Amazon MQ 現在支援 RabbitMQ 3.8.11。如需此更新中包含的修正和功能詳細資訊，請參閱下列項目：<br><br>• RabbitMQ 伺服器儲存庫上的發行說明 GitHub<br>• RabbitMQ 變更記錄<br>• 管理 Amazon MQ for RabbitMQ 引擎版本 |
| 2021 年 4 月 1 日 | Amazon MQ 現已在亞太區域 (大阪) 區域提供。如需可用區域的相關資訊，請參閱 Amazon MQ 區域與端點。 |

| 日期 | 文件更新 |
| --- | --- |
| 2020 年 12 月 21 日 | Amazon MQ 現在支援 ActiveMQ 5.15.14。如需詳細資訊，請參閱下列內容：<br>• ActiveMQ 5.15.14 版本備註<br>• 管理 Amazon MQ for ActiveMQ 引擎版本<br>• 使用 Spring XML 組態檔案<br>• **Important**<br>由於此版本中已知的 Apache ActiveMQ 問題，ActiveMQ Web 主控台中新的暫停佇列按鈕無法搭配 Amazon MQ for ActiveMQ 代理程式使用。如需此問題的詳細資訊，請參閱 AMQ-8104。 |
| 2020 年 11 月 4 日 | Amazon MQ 現在支援 RabbitMQ，這是熱門的開源訊息代理程式。這使您可以將現有的 RabbitMQ 消息代理程序遷移到， AWS 而無需重寫代碼。<br>Amazon MQ for RabbitMQ 可管理個別和叢集式訊息代理程式，並可處理佈建基礎設施、設定代理程式和更新軟體等任務。<br>• Amazon MQ 支援 RabbitMQ 3.8.6。如需支援的引擎版本詳細資訊，請參閱 the section called “版本管理”。<br>• AWS 免費方案包含最多 750 小時的單一執行個體 `mq.t3.micro` 代理程式，以及一年每個月高達 20GB 的儲存體。如需支援的執行個體類型詳細資訊，請參閱 Broker instance types。<br>• 透過 Amazon MQ for RabbitMQ，您可使用 AMQP 0-9-1 存取您的代理程式，並使用 RabbitMQ 用戶端程式庫支援的任何語言。如需支援的通訊協定和密碼套件詳細資訊，請參閱 the section called “Amazon MQ for RabbitMQ 通訊協定”。<br>• Amazon MQ for RabbitMQ 適用於目前可使用 Amazon MQ 的所有區域。若要進一步了解所有可用區域，請參閱 AWS 區域表格。<br>若要開始使用 Amazon MQ、建立代理程式，並將 JVM 型應用程式連線到 RabbitMQ 代理程式，請參閱 the section called “建立並連線至 RabbitMQ 代理程式”。 |

| 日期 | 文件更新 |
| --- | --- |
| 2020 年 10 月 22 日 | Amazon MQ 支援 ActiveMQ 5.15.13。如需詳細資訊，請參閱下列內容：<br>• ActiveMQ 5.15.13 版本備註<br>• 管理 Amazon MQ for ActiveMQ 引擎版本<br>• 使用 Spring XML 組態檔案 |
| 2020 年 9 月 30 日 | Amazon MQ 現已在歐洲 (米蘭) 區域提供。如需可用區域的相關資訊，請參閱 Amazon MQ 區域與端點。 |
| 2020 年 7 月 27 日 | 您可以使用儲存在 Active Directory 或其他 LDAP 伺服器中的登入資料來驗證 Amazon MQ 使用者。您也可以新增、刪除和修改 Amazon MQ 使用者，並指派主題和佇列的許可。如需詳細資訊，請參閱 整合 LDAP 與 ActiveMQ。 |
| 2020 年 7 月 17 日 | Amazon MQ 現在支援 `mq.t3.micro` 執行個體類型。如需詳細資訊，請參閱 Broker instance types。 |
| 2020 年 6 月 30 日 | Amazon MQ 支援 ActiveMQ 5.15.12。如需詳細資訊，請參閱下列內容：<br>• ActiveMQ 5.15.12 版本備註<br>• 管理 Amazon MQ for ActiveMQ 引擎版本<br>• 使用 Spring XML 組態檔案 |
| 2020 年 4 月 30 日 | Amazon MQ 支援 `broker` 元素上的新子集合元素 `systemUsage` 。如需詳細資訊，請參閱 systemUsage。<br><br>Amazon MQ 也支援 `kahaDB` 子元素上的三個新屬性。<br>• `journalDiskSyncInterval` - 若為 `journalDiskSyncStrategy=periodic` ，要在何時執行磁碟同步的間隔 (毫秒)。<br>• `journalDiskSyncStrategy` - 設定磁碟同步政策。<br>• `preallocationStrategy` - 設定代理程式將如何嘗試在需要新日誌檔案時，預先配置日誌檔案。<br><br>如需詳細資訊，請參閱 屬性。 |

| 日期 | 文件更新 |
| --- | --- |
| 2020 年 3 月 3 日 | Amazon MQ 支援兩個新 CloudWatch 指標<br>• `TempPercentUsage` - 非持久性訊息使用的可用臨時儲存百分比。<br>• `JobSchedulerStorePercentUsage` - 任務排程器存放區使用的磁碟空間百分比。<br><br>如需詳細資訊，請參閱 Monitoring Amazon MQ using CloudWatch。 |
| 2020 年 2 月 4 日 | Amazon MQ 已在亞太區域 (香港) 和中東 (巴林) 區域提供。如需可用區域的相關資訊，請參閱 AWS 區域與端點。 |
| 2020 年 1 月 22 日 | Amazon MQ 支援 ActiveMQ 5.15.10。如需詳細資訊，請參閱下列內容：<br>• ActiveMQ 5.15.10 版本備註<br>• 管理 Amazon MQ for ActiveMQ 引擎版本<br>• 使用 Spring XML 組態檔案 |
| 2019 年 12 月 19 日 | Amazon MQ 已在歐洲 (斯德哥爾摩) 和南美洲 (聖保羅) 區域提供。如需可用區域的相關資訊，請參閱 AWS 區域與端點。 |

| 日期 | 文件更新 |
| --- | --- |
| 2019 年 12 月 16 日 | Amazon MQ 支援使用 Amazon Elastic Block Store (EBS) 建立輸送量最佳化的代理程式，而不是適用於代理程式儲存的預設 Amazon Elastic File System (Amazon EFS)。若要利用跨多個可用區域的高耐久性和複寫功能，請使用 Amazon EFS。若要利用低延遲和高輸送量，請使用 Amazon EBS。<br><br>**Important**<br>• 您只能將 Amazon EBS 搭配 `mq.m5` 代理程式執行個體類型系列使用。<br>• 雖然您可以變更代理程式執行個體類型，但無法在建立代理程式後變更代理程式儲存類型。<br>• Amazon EBS 會複寫單一可用區域內的資料，且不支援 ActiveMQ 作用中/待命部署模式。<br><br>如需詳細資訊，請參閱下列內容：<br>• Storage<br>• 為最佳輸送量選擇正確的代理程式儲存類型<br>• Amazon MQ REST API 參考中 `broker-instance-options` 資源的 `storageType` 屬性<br>• Amazon MQ for ActiveMQ 指標 章節中的 `BurstBalance` 、`VolumeReadOps` 和 `VolumeWriteOps` 指標。 |
| 2019 年 10 月 18 日 | 有兩個 Amazon CloudWatch 指標可用：`TotalEnqueueCount` 和`TotalDequeueCount` 。如需詳細資訊，請參閱 ActiveMQ 目的地 (佇列和主題) 指標。 |

| 日期 | 文件更新 |
| --- | --- |
| 2019 年 10 月 11 日 | Amazon MQ 現在支援美國商業區域中的聯邦資訊處理標準 140-2 (FIPS) 相容端點。<br><br>如需詳細資訊，請參閱下列內容：<br><br>• 美國聯邦資訊處理標準 (FIPS) 140-2<br>• Amazon MQ 區域與端點 |
| 2019 年 9 月 30 日 | Amazon MQ 現在包含透過變更主機執行個體類型來擴展代理程式的功能。如需詳細資訊，請參閱 `UpdateBrokerInput` 的 `hostInstanceType` 屬性和 `DescribeBrokerOutput` 的 `pendingHostInstanceType` 屬性。 |
| 2019 年 8 月 30 日 | 您現在可以在主控台中更新與代理程式建立關聯的安全群組，以及使用 `UpdateBrokerInput` 更新該群組。 |
| 2019 年 7 月 22 日 | Amazon MQ 與 AWS Key Management Service (KMS) 整合以提供伺服器端加密。您現在可以選取自己的客戶管理 CMK，或在您的 AWS KMS 帳戶中使用 AWS 受管 KMS 金鑰。如需詳細資訊，請參閱 靜態加密。<br><br>Amazon MQ 支援以下列方式使用 AWS KMS 金鑰。<br><br>• AWS 擁有的 KMS 金鑰 — 金鑰擁有 Amazon MQ，不在您的帳戶中。<br>• AWS 受管 KMS 金鑰 — AWS 受管 KMS 金鑰 (aws/mq) 是您帳戶中的 KMS 金鑰，由 Amazon MQ 代表您建立、管理和使用。<br>• 選取現有客戶管理的 CMK — 客戶管理的 CMK 是您在 AWS Key Management Service (KMS) 中建立和管理的 CMK。 |
| 2019 年 6 月 19 日 | Amazon MQ 已在歐洲 (巴黎) 和亞太區域 (孟買) 區域提供。如需可用區域的相關資訊，請參閱 AWS 區域與端點。 |
| 2019 年 6 月 12 日 | Amazon MQ 已在加拿大 (中部) 區域提供。如需可用區域的相關資訊，請參閱 AWS 區域與端點。 |

| 日期 | 文件更新 |
| --- | --- |
| 2019 年 6 月 3 日 | 有兩個新的 Amazon CloudWatch 指標可用：`EstablishedConnectionsCount` 和 `InactiveDurableSubscribers`。如需詳細資訊，請參閱下列內容：<br>• Monitoring Amazon MQ using CloudWatch<br>• Amazon MQ for ActiveMQ 指標 |
| 2019 年 5 月 10 日 | 全新 `mq.t2.micro` 執行個體類型適用的資料儲存體現在限制為 20 GB。如需詳細資訊，請參閱下列內容：<br>• the section called “資料儲存體”<br>• Broker instance types |
| 2019 年 4 月 29 日 | 您現在可以使用以標籤為基礎的政策和資源級許可。如需詳細資訊，請參閱下列內容：<br>• Amazon MQ 如何搭配 IAM 運作<br>• Amazon MQ API 動作的資源層級許可 |
| 2019 年 4 月 16 日 | 您現在可以使用 REST API 擷取與代理程式引擎和代理程式執行個體選項相關的資訊。如需詳細資訊，請參閱下列內容：<br>• 代理程式執行個體選項<br>• 代理程式引擎類型 |
| 2019 年 4 月 8 日 | Amazon MQ 支援 ActiveMQ 5.15.9。如需詳細資訊，請參閱下列內容：<br>• ActiveMQ 5.15.9 版本備註<br>• 管理 Amazon MQ for ActiveMQ 引擎版本<br>• 使用 Spring XML 組態檔案 |

| 日期 | 文件更新 |
| --- | --- |
| 2019 年 3 月 4 日 | 改善有關設定動態容錯移轉和代理程式網路用戶端重新平衡的文件。設定 `transportConnectors` 及搭配 `networkConnectors` 組態選項，啟用動態容錯移轉。如需詳細資訊，請參閱下列內容：<br>• 搭配傳輸連接器的動態容錯移轉<br>• Amazon MQ 代理程式網路<br>• Amazon MQ Broker Configuration Parameters |
| 2019 年 2 月 27 日 | 除了下列區域以外，Amazon MQ 也在歐洲 (倫敦) 區域提供：<br>• 亞太區域 (新加坡)<br>• 美國東部 (俄亥俄)<br>• 美國東部 (維吉尼亞北部)<br>• 美國西部 (加利佛尼亞北部)<br>• 美國西部 (奧勒岡)<br>• 亞太區域 (東京)<br>• 亞太區域 (首爾)<br>• 亞太區域 (悉尼)<br>• 歐洲 (法蘭克福)<br>• 歐洲 (愛爾蘭) |
| 2019 年 1 月 24 日 | 預設組態現在包含清除非作用中目的地的政策。 |
| 2019 年 1 月 17 日 | Amazon MQ `mq.t2.micro` 執行個體類型目前僅支援每個線路通訊協定 100 個連線。如需詳細資訊，請參閱 Quotas in Amazon MQ。 |

| 日期 | 文件更新 |
| --- | --- |
| 2018 年 12 月 19 日 | 您可以在代理程式網路中，設定一系列的 Amazon MQ 代理程式。如需詳細資訊，請參閱下列章節：<br>• Amazon MQ 代理程式網路<br>• Creating and Configuring a Network of Brokers<br>• 正確地設定您的代理程式網路<br>• networkConnector<br>• networkConnectionStartAsync |
| 2018 年 12 月 11 日 | Amazon MQ 支援 ActiveMQ 5.15.8、5.15.6 和 5.15.0。<br>• ActiveMQ 文件中已解決的錯誤和改進項目：<br>  • ActiveMQ 5.15.8 版本備註<br>  • ActiveMQ 5.15.7 版本備註 |
| 2018 年 12 月 5 日 | AWS 支援資源標記，協助追蹤成本分配。您可以透過建立資源或檢視資源的詳細資訊，來為其加上標籤。如需詳細資訊，請參閱標記資源。 |
| 2018 年 11 月 19 日 | AWS 已擴大其 SOC 合規計畫，將 Amazon MQ 納入 SOC 合規服務。 |
| 2018 年 10 月 15 日 | • 每個使用者的群組數上限為 20。如需詳細資訊，請參閱 使用者。<br>• 根據線路通訊協定，每個代理程式的連線數上限為 1,000。如需詳細資訊，請參閱 中介裝置。 |
| 2018 年 10 月 2 日 | AWS 已擴大其 HIPAA 合規計劃，將 Amazon MQ 納入 H IPAA 合格服務。 |

| 日期 | 文件更新 |
| --- | --- |
| 2018 年 9 月 27 日 | 除了 ActiveMQ 5.15.0 外，Amazon MQ 還支援 5.15.6。如需詳細資訊，請參閱下列內容：<br>• 編輯代理程式引擎版本、執行個體類型、CloudWatch Logs，以及維護偏好設定<br>• ActiveMQ 文件中已解決的錯誤和改善：<br>  • ActiveMQ 5.15.6 版本備註<br>  • ActiveMQ 5.15.5 版本備註<br>  • ActiveMQ 5.15.4 版本備註<br>  • ActiveMQ 5.15.3 版本備註<br>  • ActiveMQ 5.15.2 版本備註<br>  • ActiveMQ 5.15.1 版本備註<br>• ActiveMQ Client 5.15.6 |
| 2018 年 8 月 31 日 | • 下列指標可供使用：<br>  • `CurrentConnectionsCount`<br>  • `TotalConsumerCount`<br>  • `TotalProducerCount`<br>  如需詳細資訊，請參閱 Amazon MQ for ActiveMQ 指標 一節。<br>• 代理程式的 IP 地址會顯示在詳細資訊頁面上。<br>  **Note**<br>  對於公開存取性已停用的代理程式，會顯示內部 IP 地址。 |

| 日期 | 文件更新 |
| --- | --- |
| 2018 年 8 月 30 日 | 除了下列區域以外，Amazon MQ 也在亞太區域 (新加坡) 區域提供：<br>• 美國東部 (俄亥俄)<br>• 美國東部 (維吉尼亞北部)<br>• 美國西部 (加利佛尼亞北部)<br>• 美國西部 (奧勒岡)<br>• 亞太區域 (東京)<br>• 亞太區域 (首爾)<br>• 亞太區域 (悉尼)<br>• 歐洲 (法蘭克福)<br>• 歐洲 (愛爾蘭) |
| 2018 年 7 月 30 日 | 您可以將 Amazon MQ 設定為將一般日誌和稽核日誌發佈到 Amazon CloudWatch 日誌。如需詳細資訊，請參閱 Configuring Amazon MQ to publish logs to Amazon CloudWatch Logs。 |
| 2018 年 7 月 25 日 | 除了下列區域以外，Amazon MQ 也在亞太區域 (東京)與亞太區域 (首爾) 區域提供：<br>• 美國東部 (俄亥俄)<br>• 美國東部 (維吉尼亞北部)<br>• 美國西部 (加利佛尼亞北部)<br>• 美國西部 (奧勒岡)<br>• 亞太區域 (悉尼)<br>• 歐洲 (法蘭克福)<br>• 歐洲 (愛爾蘭) |
| 2018 年 7 月 19 日 | 您可以使 AWS CloudTrail 用記錄 Amazon MQ API 呼叫。如需詳細資訊，請參閱 Logging Amazon MQ API calls using CloudTrail。 |

| 日期 | 文件更新 |
| --- | --- |
| 2018 年 6 月 29 日 | 除了 `mq.t2.micro` 和 `mq.m4.large` 之外，以下代理程式執行個體類型也適用於一般開發、測試和生產工作負載，而這些工作負載需要高傳輸量：<br>• `mq.m5.large`<br>• `mq.m5.xlarge`<br>• `mq.m5.2xlarge`<br>• `mq.m5.4xlarge`<br>如需詳細資訊，請參閱 Broker instance types。 |
| 2018 年 6 月 27 日 | 除了下列區域以外，Amazon MQ 也在美國西部 (加利佛尼亞北部) 區域提供：<br>• 美國東部 (俄亥俄)<br>• 美國東部 (維吉尼亞北部)<br>• 美國西部 (奧勒岡)<br>• 亞太區域 (悉尼)<br>• 歐洲 (法蘭克福)<br>• 歐洲 (愛爾蘭) |

| 日期 | 文件更新 |
| --- | --- |
| 2018 年 6 月 14 日 | • 您可以使用AWS::Amazon MQ::Broker AWS CloudFormation 資源執行下列動作：<br>  • 建立代理程式。<br>  • 新增組態變更或修改所指定代理程式的使用者。<br>  • 傳回所指定代理程式的相關資訊。<br>  • 刪除指定的代理程式。<br><br>  Note<br>  當您變更 Amazon MQ 代理程式 ConfigurationId或 Amazon MQ 代理程式使用者屬性類型的任何屬性時，代理程式會立即重新啟動。<br>• 您可以使用AWS::Amazon MQ::Configuration AWS CloudFormation 資源執行下列動作：<br>  • 建立組態。<br>  • 更新指定的組態。<br>  • 傳回所指定組態的相關資訊。<br><br>  Note<br>  您可以使用修 AWS CloudFormation 改 (但不能刪除) Amazon MQ 組態。 |
| 2018 年 6 月 7 日 | Amazon MQ 主控台支援德文、巴西葡萄牙文、西班牙文、義大利文和繁體中文。 |
| 2018 年 5 月 17 日 | 每個代理程式的使用者數目限制為 250。如需詳細資訊，請參閱 使用者。 |
| 2018 年 3 月 13 日 | 建立代理程式大約需要 15 分鐘。如需詳細資訊，請參閱完成代理程式的建立。 |

| 日期 | 文件更新 |
| --- | --- |
| 2018 年 3 月 1 日 | • 您可以使用 `concurrentStoreAndDispatchQueues` 屬性來設定 Apache KahaDB 的並行存放和分派。<br>• 此`CpuCreditBalance` CloudWatch 測量結果適用於中`mq.t2.micro` 介執行處理類型。 |
| 2018 年 1 月 10 日 | 以下變更會影響 Amazon MQ 主控台：<br>• 在代理程式清單中，預設會隱藏 Creation (建立) 欄。若要自訂頁面大小和資料欄，請選擇 。<br>• 在***MyBroker***頁面的 [連線] 區段中，選擇安全群組的名稱或開啟 EC2 主控台 (而非 VPC 主控台)。EC2 主控台可讓您更直覺地設定傳入和傳出規則。如需詳細資訊，請參閱更新的啟用傳入連線章節。 |
| 2018 年 1 月 9 日 | • REST 操作 ID `UpdateBroker` 的許可在 IAM 主控台上正確地列示為 `mq:UpdateBroker` 。<br>• 錯誤的 `mq:DescribeEngine` 許可會從 IAM 主控台中移除。 |

| 日期 | 文件更新 |
| --- | --- |
| 2017 年 11 月 28 日 | 這是 Amazon MQ 和 Amazon MQ 開發人員指南的最初版本。<br>• Amazon MQ 已在下列區域提供 ：<br>  • 美國東部 (俄亥俄)<br>  • 美國東部 (維吉尼亞北部)<br>  • 美國西部 (奧勒岡)<br>  • 亞太區域 (悉尼)<br>  • 歐洲 (法蘭克福)<br>  • 歐洲 (愛爾蘭)<br>  `mq.t2.micro` 執行個體類型的使用受限於 CPU 額度和基準效能—並能夠大幅提升 效能以超越基準水準 (如需詳細資訊，請參閱 `CpuCreditBalance` 指標)。如果您的應用程式需要固定效能，請考慮使用 `mq.m5.large` 執行個體類型。<br>• 您可以建立 `mq.m4.large` 和 `mq.t2.micro` 代理程式。<br>  `mq.t2.micro` 執行個體類型的使用受限於 CPU 額度和基準效能—並能夠大幅提升效能以超越基準水準 (如需詳細資訊，請參閱 `CpuCreditBalance` 指標)。如果您的應用程式需要固定效能，請考慮使用 `mq.m5.large` 執行個體類型。<br>• 您可以使用 ActiveMQ 5.15.0 代理程式引擎。<br>• 您也可以使用 Amazon MQ REST API 和 AWS 開發套件，以程式設計方式建立和管理代理程式。<br>• 您可以存取代理程式，方法為使用 ActiveMQ 支援的任何程式設計語言，並明確地為下列通訊協定啟用 TLS：<br>  • AMQP<br>  • MQTT<br>  • MQTT 超過 WebSocket<br>  • OpenWire<br>  • STOMP<br>  • 跺腳過來 WebSocket |

| 日期 | 文件更新 |
| --- | --- |
| | • 您可以使用多種 ActiveMQ 用戶端連線至 ActiveMQ 代理程式。建議使用 ActiveMQ 用戶端。如需詳細資訊，請參閱 Connecting a Java application to your broker。<br>• 您的代理程式可以傳送和接收任何大小的訊息。 |

# Amazon MQ 文件歷史記錄

下表列出 Amazon MQ 開發人員指南的變更。如需 Amazon MQ 功能發佈與改進的資訊，請參閱 Amazon MQ 版本備註。

| 日期 | 文件更新 |
| --- | --- |
| 2022 年 8 月 22 日 | 為 Amazon MQ for ActiveMQ 和 Amazon MQ for RabbitMQ 引擎建立個別的父章節。這些父章節現在包含引擎詳細資料、教學課程和最佳實務：<br>• Working with Amazon MQ for ActiveMQ<br>• Working with Amazon MQ for RabbitMQ |
| 2022 年 1 月 13 日 | 新增了新的疑難排解區段，其中列出了代理程式處於運作不佳狀態時 Amazon MQ 傳回的狀態碼，還包括有關診斷、復原代理程式的詳細資訊。<br>• the section called “故障診斷：需執行的 Amazon MQ 動作的代碼” |
| 2021 年 11 月 8 日 | 新增了新的教學課程，說明如何使用 Amazon MQ for RabbitMQ 代理程式來設定 Python Pika 用戶端。<br>• the section called “將 Python Pika 與 Amazon MQ for RabbitMQ 搭配使用” |
| 2021 年 10 月 8 日 | 已為 Amazon MQ for ActiveMQ 和 Amazon MQ for RabbitMQ 代理程式引擎新增下列疑難排解主題：<br>• 有些用戶端無法連線<br>• the section called “如何在 Amazon MQ for RabbitMQ 中啟用外掛程式？”<br>• the section called “我無法變更代理程式的 Amazon VPC 組態。” |

| 日期 | 文件更新 |
| --- | --- |
| 2021 年 9 月 22 日 | 已新增下列主題，以便對 Amazon MQ for ActiveMQ 代理程式的常見連線和授權問題進行疑難排解：<br>• 重新啟動後連線到代理程式<br>• Web 主控台上的 JSP 例外狀況 |
| 2021 年 8 月 12 日 | 新增以下章節，說明使用 Amazon MQ 代理程式時常見問題的疑難排解。<br>• 故障診斷 |
| 2021 年 7 月 29 日 | 新增下列章節，說明 Amazon MQ for RabbitMQ 版本管理，並將 Amazon MQ 代理程式升級至新的次要和主要引擎版本 (如受支援)。<br>• the section called “版本管理” |
| 2021 年 7 月 21 日 | 新增以下各節，說明將 Amazon MQ 代理程式連接 AWS Lambda 為事件來源。<br>• Connect your Amazon MQ for ActiveMQ broker to Lambda<br>• Connect your Amazon MQ for RabbitMQ broker to Lambda |
| 2021 年 7 月 16 日 | 已新增下列各節，說明 Amazon MQ 代理程式維護時段，以及如何使用 AWS Management Console AWS CLI、或 Amazon MQ API 調整維護時段。<br>• the section called “維護代理程式” |
| 2021 年 6 月 7 日 | 新增下列章節，說明 Amazon MQ for ActiveMQ 版本管理，並將 Amazon MQ 代理程式升級至新的次要和主要引擎版本 (如受支援)。<br>• the section called “版本管理”<br>• the section called “升級引擎版本” |
| 2021 年 5 月 18 日 | 新增以下章節，說明 Amazon MQ for RabbitMQ 代理程式預設值<br>• the section called “代理程式預設值” |

| 日期 | 文件更新 |
| --- | --- |
| 2021 年 5 月 5 日 | 已新增下列章節，說明 Amazon MQ 的 AWS 受管政策以及這些政策的更新：<br>• the section called “AWS 受管政策” |
| 2021 年 2 月 16 日 | 為 Amazon MQ for RabbitMQ 新增以下教學章節：<br>• the section called “解決暫停的佇列同步” |
| 2020 年 11 月 4 日 | • 新增下列章節以記載 Amazon MQ for RabbitMQ 支援：<br>  • the section called “建立並連線至 RabbitMQ 代理程式”<br>  • the section called “RabbitMQ 教學”<br>  • the section called “Amazon MQ for RabbitMQ 最佳實踐”<br>  • the section called “RabbitMQ 引擎”<br>  • the section called “設定 Amazon MQ for RabbitMQ 日誌”<br>  • the section called “使用服務連結角色”<br>• 對本指南現有章節進行其他修訂，以準確記載 Amazon MQ for RabbitMQ 支援。 |
| 2019 年 12 月 16 日 | • 新增下列章節：<br>  • Storage<br>  • 為最佳輸送量選擇正確的代理程式儲存類型<br>• 修訂以下章節中的資訊：<br>  • 中介裝置<br>  • Broker instance types<br>  • Amazon MQ 單一執行個體代理程式<br>  • 高可用性的 Amazon MQ 作用中/待命代理程式<br>  • Create an ActiveMQ broker<br>  • Creating and configuring a broker |

| 日期 | 文件更新 |
| --- | --- |
| 2019 年 7 月 19 日 | 下列章節中已修改及新增有關加密管理的內容：<br>• Amazon MQ 的資料保護<br>  • 靜態加密<br>  • 傳輸中加密<br>• `EncryptionOptions` |
| 2019 年 4 月 22 日 | 針對以標籤為基礎的政策和資源層級許可已新增以下章節：<br>• Amazon MQ 如何搭配 IAM 運作<br>• Amazon MQ API 動作的資源層級許可 |
| 2019 年 3 月 4 日 | 改善有關設定動態容錯移轉和代理程式網路用戶端重新平衡的文件。設定 `transportConnectors` 及搭配 `networkConnectors` 組態選項，啟用動態容錯移轉。<br>• 搭配傳輸連接器的動態容錯移轉<br>• Amazon MQ 代理程式網路<br>• Amazon MQ Broker Configuration Parameters |
| 2019 年 1 月 5 日 | 改善部分每分鐘指標的文件。如需詳細資訊，請參閱下列內容：ActiveMQ 目的地 (佇列和主題) 指標。 |
| 2018 年 12 月 19 日 | • 新增下列章節：<br>  • Amazon MQ 代理程式網路<br>  • Creating and Configuring a Network of Brokers<br>  • 正確地設定您的代理程式網路<br>  • `networkConnector`<br>  • `networkConnectionStartAsync`<br>• 已將 `networkConnectors` 子集合元素加入 Amazon MQ 組態中允許的元素、子集合元素及其子元素 區塊。 |
| 2018 年 12 月 11 日 | 已更新文件，以顯示最新的 ActiveMQ version 5.15.8 可用性。 |

| 日期 | 文件更新 |
| --- | --- |
| 2018 年 12 月 5 日 | 新增標記 資源章節。 |
| 2018 年 10 月 26 日 | 新增透過復原備妥的 XA 交易避免緩慢重新啟動章節。 |
| 2018 年 10 月 15 日 | 更新Quotas in Amazon MQ章節。 |
| 2018 年 10 月 1 日 | 更正後續步驟章節中的資訊。 |
| 2018 年 9 月 27 日 | • 新增編輯代理程式引擎版本、執行個體類型、CloudWatch Logs，以及維護偏好設定章節。<br>• 已更新下列各節：<br>  • Create an ActiveMQ broker<br>  • Configure Basic Broker Settings |
| 2018 年 9 月 18 日 | 將以下附註新增到 建立和管理 ActiveMQ 代理程式使用者 章節：您無法獨立於使用者設定群組。當您至少新增一位使用者至群組標籤時，就會建立群組標籤，並在您移除其中的所有使用者時刪除該群組標籤。 |
| 2018 年 8 月 31 日 | • 已釐清作用中/待命代理程式的術語。如需詳細資訊，請參閱 高可用性的 Amazon MQ 作用中/待命代理程式。<br>• 已簡化維護時段的術語。如需詳細資訊，請參閱 Amazon MQ 代理程式組態生命週期。<br>• 重寫Configure Additional Broker Settings章節。<br>• 已更新Amazon MQ for ActiveMQ 指標和Listing brokers and viewing broker details章節。 |
| 2018 年 8 月 15 日 | 更正Create an ActiveMQ broker章節中的資訊。 |
| 2018 年 8 月 13 日 | 新增在沒有公開存取性的情況下存取代理程式 Web 主控台章節。 |

| 日期 | 文件更新 |
| --- | --- |
| 2018 年 8 月 2 日 | • 新增CloudWatch Logs 組態疑難排解章節。<br>• 已將以下警告新增到整本指南：<br>**Important**<br>在以下的範例程式碼中，生產者和消費者會在單一執行緒中執行。對於生產系統 (或測試代理程式執行個體容錯移轉)，請確定您的生產者和消費者在不同的主機或執行緒上執行。 |
| 2018 年 8 月 1 日 | 更正以下章節中的資訊：<br>• 了解 CloudWatch Logs 中的記錄結構<br>• Connect a Java application to your broker |
| 2018 年 7 月 31 日 | • 已將 3 分鐘示範影片移至Getting Started with Amazon MQ章節。<br>• 已將 3 分鐘入門影片移至What is Amazon MQ?章節。 |
| 2018 年 7 月 30 日 | • 新增Configuring Amazon MQ to publish logs to Amazon CloudWatch Logs章節。<br>• 更新Configure Additional Broker Settings章節。 |
| 2018 年 7 月 19 日 | • 新增Logging Amazon MQ API calls using CloudTrail章節。 |
| 2018 年 7 月 5 日 | • 已將 `authorizationEntry` 子元素交叉參考新增至一律設定授權映射章節。<br>• 釐清整合 ActiveMQ 代理程式與 LDAP章節中的資訊。<br>• 釐清API 調節章節中的資訊。 |
| 2018 年 6 月 29 日 | • 已更新Broker instance types章節中的資訊。<br>• 新增為最佳傳輸量選擇正確的代理程式執行個體類型章節。 |

| 日期 | 文件更新 |
| --- | --- |
| 2018 年 4 月 6 日 | 除了 HTML GitHub、PDF 和 Kindle 之外，Amazon MQ 開發人員指南發行說明也可以作為 RSS 摘要提供。  |
| 2018 年 5 月 29 日 | 已在Working Java Example章節中進行下列變更：<br>• 已新增 STOMP+WSS Java 範例。STOMP+WSS 範例 Java 程式碼會連接至代理程式、建立主題，以及發佈和接收訊息。<br>• 已改善 MQTT Java 範例。<br>• 改進了 OpenWire Java 的例子。 |
| 2018 年 5 月 24 日 | 已在Working Java Example章節更正 MQTT Java 範例中的線路通訊協定端點連接埠。 |
| 2018 年 5 月 22 日 | 已更正所有 Java 相依性部分中的資訊。 |
| 2018 年 5 月 17 日 | 更正使用者章節中的資訊。 |
| 2018 年 5 月 15 日 | 更正確保有效的 Amazon MQ 效能章節中的資訊。 |
| 2018 年 5 月 8 日 | • 已將Amazon MQ REST API 許可參考放在其專屬章節。<br>• 已建立 建立 Amazon MQ 代理程式所需的 IAM 許可 章節，其中具有範例自訂 IAM 政策。 |
| 2018 年 5 月 7 日 | • 已將在整本指南中的代理程式維護時段釐清為 2 小時。如需詳細資訊，請參閱 Amazon MQ 代理程式組態生命週期。<br>• 已新增為何建立代理程式需要 `ec2:CreateNetworkInterface` 和 `ec2:CreateNetworkInterfacePermission` 許可的說明。如需詳細資訊，請參閱 Amazon MQ 的 API 身分驗證和授權。 |

| 日期 | 文件更新 |
| --- | --- |
| 2018 年 5 月 1 日 | 已在下列各節釐清作用中/待命代理程式之維護時段的相關資訊：<br>• 高可用性的 Amazon MQ 作用中/待命代理程式<br>• Creating and configuring a broker<br>• Creating and applying broker configurations |
| 2018 年 4 月 27 日 | 已重新撰寫以下各節並最佳化範例 Java 程式碼，以符合將連線集區僅用於生產者而非消費者的建議：<br>• 一律使用連線集區<br>• 建立訊息生產者和傳送訊息<br>• 建立訊息消費者和接收訊息<br>• AmazonMQExample.java |
| 2018 年 4 月 26 日 | 已將 MQTT Java 範例新增至 Working Java Example 章節。MQTT 範例 Java 程式碼會連接至代理程式、建立主題，以及發佈和接收訊息。 |
| 2018 年 4 月 4 日 | 已將與 Amazon MQ 通訊章節重新命名為 連結至 Amazon MQ。 |
| 2018 年 4 月 3 日 | 釐清並更正 對於具有緩慢消費者的佇列，停用並行存放和分派 區段中的資訊。 |
| 2018 年 4 月 2 日 | 已將 Amazon MQ 章節中佇列的並行存放和分派移至章節 對於具有緩慢消費者的佇列，停用並行存放和分派。 |
| 2018 年 3 月 27 日 | • 已使用 3 分鐘示範影片取代 What is Amazon MQ? 章節中的 re:Invent 推出影片。<br>• 重組以下章節：<br>  • Broker Architecture<br>  • Amazon MQ 的運作方式。<br>• 已移動Broker Architecture章節下的Amazon MQ 代理程式組態生命週期。 |

| 日期 | 文件更新 |
| --- | --- |
| 2018 年 3 月 22 日 | 在本指南中釐清下列陳述：Amazon MQ 使用其安全管理和儲存的加密金鑰來加密靜態和傳輸中的訊息。如需詳細資訊，請參閱《AWS Encryption SDK 開發人員指南》。 |
| 2018 年 3 月 19 日 | 在本指南中釐清下列陳述：作用中/待命代理程式是由兩個不同可用區域中的兩個代理程式所組成，並設定於備援組合中。這些代理程式會與您的應用程式及 Amazon EFS 同步通訊。 |
| 2018 年 3 月 15 日 | • 重組Amazon MQ Basic elements章節。 |
| 2018 年 3 月 12 日 | • 釐清並更正Amazon MQ 的安全最佳實務和連結至 Amazon MQ章節中的資訊。<br>• 新增對於具有緩慢消費者的佇列，停用並行存放和分派章節。<br>• 已將警告組成設定進階代理程式設定章節的前言。 |
| 2018 年 3 月 9 日 | • 釐清並更正 一律設定授權映射 區段中的資訊。<br>• 已新增authorizationEntry章節並更新kahaDB章節。 |
| 2018 年 3 月 8 日 | • 新增一律設定授權映射章節。<br>• 已將代理程式尾碼的注意事項新增至 Monitoring Amazon MQ using CloudWatch章節。 |
| 2018 年 3 月 6 日 | 新增以下附註到整本指南：<br>**Note**<br>`mq.t2.micro` 執行個體類型的使用受限於 CPU 額度和基準效能—並能夠大幅提升效能以超越基準水準 (如需詳細資訊，請參閱 `CpuCreditBalance` 指標)。如果您的應用程式需要固定效能，請考慮使用 `mq.m5.large` 執行個體類型。 |

| 日期 | 文件更新 |
| --- | --- |
| 2018 年 3 月 1 日 | • 已將 `CpuCreditBalance` 指標新增至Amazon MQ for ActiveMQ 指標章節。<br>• 新增Amazon MQ 子元素屬性章節。<br>• 已將 the section called “允許的元素” 章節中的連結來源元素新增至它們的屬性，以及新增至子集合元素。<br>• 對中的 AWS 詞彙進行了更正 GitHub。 |
| 2018 年 2 月 28 日 | 已校正影像顯示 GitHub。 |
| 2018 年 2 月 27 日 | 除了 HTML、PDF 和 Kindle 之外，Amazon MQ 開發人員指南也可在上取得。GitHub若要留下信用評價，請選擇右上角的 GitHub 圖示。<br>  |
| 2018 年 2 月 26 日 | • 已讓區域在所有範例和圖表中保持一致。<br>• 最佳化 AWS 主控台和產品網頁的連結。 |
| 2018 年 2 月 22 日 | 已釐清並更正下列章節中的資訊：<br>• 偏好無法公開存取的代理程式<br>• 一律使用容錯移轉傳輸來連接到多個代理程式端點<br>• Amazon MQ 的 API 身分驗證和授權<br>• 整合 ActiveMQ 代理程式與 LDAP |
| 2018 年 2 月 21 日 | 已更正以下章節中的 Java 程式碼：<br>• Working Java Example<br>• Connect a Java application to your broker<br>• 一律使用連線集區 |
| 2018 年 2 月 20 日 | 釐清並更正 Amazon MQ 的安全性 和「最佳實務」章節中的資訊。 |

| 日期 | 文件更新 |
| --- | --- |
| 2018 年 2 月 19 日 | • 已更正一律使用連線集區章節中的 Java 程式碼。<br>• 重新架構並擴展了「最佳實務」章節和 Amazon MQ 的安全性 章節。 |
| 2018 年 2 月 16 日 | • 新增Amazon MQ 的安全最佳實務章節。<br>• 更新連結至 Amazon MQ章節。<br>• 已更正以下章節中的 Java 程式碼：<br>  • Getting Started with Amazon MQ<br>  • AmazonMQExample.java |
| 2018 年 2 月 15 日 | • 重新架構並擴展了「最佳實務」章節。<br>• 已更新下列各節：<br>  • 如何開始使用 Amazon MQ？<br>  • 後續步驟 (入門)<br>  • Related resources |
| 2018 年 2 月 14 日 | 已更新下列各節：<br>• Quotas in Amazon MQ<br>• API 調節<br>• Amazon MQ 的安全性 |
| 2018 年 2 月 13 日 | • 更新Related resources章節。<br>• 更新Quotas in Amazon MQ章節。<br>• 新增我們希望傾聽您的意見章節。 |
| 2018 年 1 月 25 日 | • 已修正Working Java Example章節的新增 Java 相依性小節中的錯誤。<br>• REST 操作 ID `RebootBroker` 的許可在 IAM 主控台上正確地列示為 `mq:RebootBroker` 。 |
| 2018 年 1 月 24 日 | • 新增永不修改或刪除 Amazon MQ 彈性網路界面章節。<br>• 已更新整本指南的所有圖表。<br>• 已將整本指南中 Amazon MQ REST API 參考的連結和特定 REST API 的連結新增至 Amazon MQ 的 API 身分驗證和授權 章節。 |

| 日期 | 文件更新 |
| --- | --- |
| 2018 年 1 月 19 日 | 已更新Amazon MQ for ActiveMQ 資源章節中的資訊。 |
| 2018 年 1 月 18 日 | 釐清並更正 Quotas in Amazon MQ 區段中的資訊。 |
| 2018 年 1 月 17 日 | 已恢復比起耐久訂閱更喜歡虛擬目的地的建議，並附上改善的說明。 |
| 2018 年 1 月 11 日 | • 除了提供 HTML 和 PDF 版本外，《Amazon MQ 開發人員指南》也以 Kindle 格式提供。<br>• 已釐清並更正Amazon MQ 的 API 身分驗證和授權和步驟 2：建立使用者並取得您的 AWS 登入資料章節中的資訊。 |
| 2018 年 1 月 3 日 | 新增`DescribeConfigurationRevision` 至Amazon MQ 的 API 身分驗證和授權章節。 |
| 2017 年 12 月 15 日 | 已從「最佳實務」章節中移除針對耐久訂閱的建議。 |
| 2017 年 12 月 8 日 | • 已將啟用傳入連線先決條件新增至Connecting a Java application to your broker和Working Java Example章節。<br>• 已在本指南中新增以下附註：您目前無法刪除組態。 |
| 2017 年 12 月 7 日 | • 已改善AmazonMQExample.java中的程式碼。<br>• 新增Amazon MQ 的 API 身分驗證和授權章節。 |
| 2017 年 12 月 5 日 | • 已釐清並更正Monitoring Amazon MQ using CloudWatch章節中的資訊：<br>  • 已改善指標描述。<br>  • 已新增Amazon MQ for ActiveMQ 指標和ActiveMQ 代理程式指標的維度小節。<br>• 已將「Amazon MQ 簡介」影片新增至 What is Amazon MQ? 章節。 |

| 日期 | 文件更新 |
| --- | --- |
| 2017 年 12 月 4 日 | • 已釐清資料儲存體章節中的以下資訊：每個代理程式的儲存容量為 200 GB。<br>• 已將先決條件新增至Working Java Example章節。(需有 `activemq-client.jar` 和 `activemq-pool.jar` 套件，範例才能運作。如需詳細資訊，請參閱 Connecting a Java application to your broker)。 |
| 2017 年 12 月 1 日 | • 已更新並改善所有教學課程中的螢幕擷取畫面。<br>• 在本指南中釐清下列說明：對組態修訂版或 ActiveMQ 使用者進行變更，不會立即套用變更。若要套用變更，您必須等待下一個維護時段或重新啟動代理程式。如需詳細資訊，請參閱 Amazon MQ 代理程式組態生命週期。 |

# AWS 詞彙表

如需最新的 AWS 術語，請參閱《AWS 詞彙表 參考》中的 AWS 詞彙表。

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。