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 Apache ActiveMQ Classic 和 RabbitMQ 的受管訊息代理程式服務，可管理訊息代理程式的設定、操作和維護。您可以使用產業標準簡訊通訊協定建立新的 Amazon MQ 代理程式，或將現有的訊息代理程式遷移至 Amazon MQ，而無需重寫簡訊程式碼。

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。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。您可以使用 Amazon MQ 代理程式在大規模、雲端原生應用程式和元件之間進行通訊。

主題

Amazon MQ 功能

受管維護和版本升級

Amazon MQ 會在排定的維護時段期間，為訊息代理程式執行維護和版本升級。

使用 CloudWatch 監控代理程式

Amazon MQ 已與 Amazon CloudWatch 整合，因此您可以檢視和分析代理程式和佇列的指標。您可以從 Amazon MQ 主控台、CloudWatch 主控台、命令列和 API 檢視和分析指標。系統每分鐘會自動收集指標並推送至 CloudWatch。

安全性

Amazon MQ 提供靜態和傳輸中訊息的加密。與代理程式的連線使用 SSL，而且存取可以限制在 Amazon VPC 內的私有端點。附加性，您可以使用 AWS Identity and Access Management(IAM) 控制 IAM 使用者和群組對特定 Amazon MQ 代理程式可採取的動作。

Amazon MQ 上 RabbitMQ 的配額佇列 Amazon MQ

配額佇列是由領導節點（主要複本）和跟隨節點（其他複本）組成的複寫佇列類型。每個節點都位於不同的可用區域，因此，如果一個節點暫時無法使用，訊息傳遞會繼續在另一個可用區域中新選擇的領導複本。配額佇列適用於處理毒訊息，當訊息失敗且多次重新排入佇列時，就會發生這些訊息。

Amazon MQ 上 ActiveMQ 的跨區域資料複寫

跨區域資料複寫 (CRDR) 允許從主要 AWS 區域中的主要代理程式非同步訊息複寫到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。

如何開始使用 Amazon MQ？

若要在 Amazon MQ 上開始使用 ActiveMQ Amazon MQ，請檢閱下列文件：

- 入門：建立並連線至 ActiveMQ 代理程式
- the section called “部署代理程式”
- ActiveMQ 教學課程
- the section called “Amazon MQ for ActiveMQ 最佳實務”

若要在 Amazon MQ 上開始使用 RabbitMQ Amazon MQ，請檢閱下列文件：

- 入門：建立並連線至 RabbitMQ 代理程式
- the section called “部署 RabbitMQ 代理程式”
- the section called “RabbitMQ 教學”
- the section called “Amazon MQ for RabbitMQ 最佳實踐”

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

若要了解 Amazon MQ AWS CLI 命令，請參閱《 AWS CLI 命令參考》中的 Amazon MQ。

如何提供意見回饋給 Amazon MQ？

我們歡迎並鼓勵您對文件的意見回饋。您可以使用右側的拇指向上和拇指向下圖示來提交意見回饋，也可以使用下方連結的「提供意見回饋」表單。

若要聯絡 Amazon MQ 團隊，請使用 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

註冊 AWS 帳戶

若要開始使用 AWS，您需要 AWS 帳戶。如需建立 的相關資訊 AWS 帳戶，請參閱《 AWS 帳戶管理參考指南》中的 入門 AWS 帳戶。

準備好使用範例程式碼

下列教學課程說明如何使用 與 Amazon MQ 代理程式搭配使用， AWS 管理主控台 以及如何以程式設計方式連接至 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 SDKs，以程式設計方式建立和管理代理程式。

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。

入門：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。中介裝置執行個體類別 (m5) 和大小 (large、medium) 的合併描述稱為中介裝置執行個體類型 （ 例如 mq.m5.large)。如需詳細資訊，請參閱什麼是 Amazon MQ for ActiveMQ 代理程式？。

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。下列範例示範如何使用 AWS 管理主控台 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 Amazon MQ for ActiveMQ 代理程式的部署選項。

      - 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。如需詳細資訊，請參閱選項 1：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。如需詳細資訊，請參閱選項 2：Amazon MQ 作用中/待命代理程式可提供高可用性。

   b. 選擇 Storage type (儲存體類型) (例如，EBS)。如需詳細資訊，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。其他 AWS 服務可存取中介裝置名稱，包括 CloudWatch Logs。代理程式名稱不適用於私有或敏感資料。

> **Note**
>
> 在其他設定區段中，您也可以設定下列項目：
>
> - 組態
> - CloudWatch 日誌
> - 私有存取
> - 中介裝置維護時段

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。

5. 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：

   - 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。其他 AWS 服務可存取中介裝置使用者名稱，包括 CloudWatch Logs。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
   ```

入門：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。中介裝置執行個體類別 (m5) 和大小 (large、medium) 的合併描述稱為中介裝置執行個體類型 （ 例如 mq.m5.large)。如需詳細資訊，請參閱什麼是 Amazon MQ for RabbitMQ 代理程式？

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。下列範例示範如何使用 AWS 管理主控台 來建立基本代理程式。

建立 Amazon MQ for RabbitMQ 代理程式時，請遵循 RabbitMQ 的代理程式設定最佳實務，以最大化代理程式效能並最佳化訊息輸送量效率。

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，請參閱選項 1：Amazon MQ for RabbitMQ 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，請參閱選項 2：Amazon MQ for RabbitMQ 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。其他 AWS 服務可存取中介裝置名稱，包括 CloudWatch Logs。代理程式名稱不適用於私有或敏感資料。

b. 選擇中介裝置執行個體類型 （ 例如，mq.m7g.large)。如需詳細資訊，請參閱Broker instance types。

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。其他 AWS 服務可存取中介裝置使用者名稱，包括 CloudWatch Logs。代理程式使用者名稱不適用於私有或敏感資料。

   > **Note**
   >
   > 在其他設定區段中，您也可以設定下列項目：
   >
   > - 組態
   > - CloudWatch 日誌
   > - 私有存取
   > - 中介裝置維護時段

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 (連線) 區段中，請記下代理程式的 RabbitMQ web console (RabbitMQ Web 主控台) URL，例如：

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

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.on.aws:5671
```

管理 Amazon MQ 代理程式

建立代理程式後，您可以管理和維護 Amazon MQ 代理程式的不同元件。

主題

連結至 Amazon MQ

您可以使用 AWS 服務端點和代理程式端點，從其他服務連線至 Amazon MQ。

服務端點

下列連線方法用於 Amazon MQ 服務 API：

網域	連線方法
`mq.region.amazonaws.com`	IPv4
`mq.region.api.aws`	雙堆疊 (IPv4 和 IPv6)
`mq-fips.region.amazonaws.com`	僅限 IPv4 的 FIPS
`mq-fips.region.api.aws`	具有雙堆疊的 FIPS

中介裝置端點

下列連線方法用於 Amazon MQ 代理程式：

網域	連線方法
brokerId.mq.*region*.amazonaws.com	IPv4
brokerId.mq.*region*.on.aws	雙堆疊 (IPv4 和 IPv6) **Note** Amazon MQ for ActiveMQ 代理程式不支援雙堆疊。

使用雙堆疊 (IPv4 和 IPv6) 端點連線至 Amazon MQ

雙堆疊端點同時支援 IPv4 和 IPv6 流量。當您向雙堆疊端點提出請求時，端點 URL 會解析為 IPv4 或 IPv6 地址。如需雙堆疊和 FIPS 端點的詳細資訊，請參閱 SDK 參考指南。

Amazon MQ 支援區域雙堆疊端點，這表示您必須將 AWS 區域指定為端點名稱的一部分。雙堆疊端點名稱使用以下命名慣例：`mq.region.api.aws`。例如，`eu-west-1` 區域的雙堆疊端點名稱是 `mq.eu-west-1.api.aws`。

如需 Amazon MQ 端點的完整清單，請參閱 AWS 一般參考。

使用 AWS PrivateLink 連線至 Amazon MQ

Amazon MQ API 的 AWS PrivateLink 端點支援 IPv4 和 IPv6，可在虛擬私有雲端 (VPCs) 和 Amazon MQ API 之間提供私有連線，而不會將您的流量暴露到公有網際網路。

Note

PrivateLink 的支援僅適用於 Amazon MQ API 端點，不適用於代理程式端點。如需私有連線至代理程式端點的詳細資訊，請參閱 Configuring a private Amazon MQ broker。

若要使用 PrivateLink 存取 Amazon MQ API，您必須先在您要連線的特定 VPC 中建立介面 VPC 端點。建立 VPC 端點時，請使用服務名稱`com.amazonaws.region.mq`或`com.amazonaws.region.mq-fips` FIPS 端點。

當您使用 CLI 或 SDK AWS 呼叫 Amazon MQ 時，您必須指定端點 URL 才能使用雙堆疊網域名稱：`mq.region.api.aws`或`mq-fips.region.api.aws`。PrivateLink for Amazon MQ 不支援以 結尾的預設網域名稱`amazonaws.com`。如需詳細資訊，請參閱 SDK 參考指南中的雙堆疊和 FIPS 端點。

下列 CLI 範例示範如何透過 Amazon MQ VPC 端點`describe-broker-engine-type`在亞太區域（雪梨）區域呼叫 。

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

如需在 CLI 中設定端點的其他方法，請參閱 CLI AWS 中的使用端點

您也可以使用 VPC 端點政策來判斷使用者對 VPC 端點的存取。如需詳細資訊，請參閱使用端點政策控制對 VPC 端點的存取。

Amazon MQ 代理程式的身分驗證和授權

Amazon MQ 提供多種身分驗證和授權方法，可根據組織的需求來保護您的訊息基礎設施。

Amazon MQ for RabbitMQ 的身分驗證和授權

Amazon MQ for RabbitMQ 支援下列身分驗證和授權方法：

簡單身分驗證和授權

在此方法中，代理程式使用者存放在 RabbitMQ 代理程式內部，並透過 Web 主控台或管理 API 進行管理。虛擬主機、交換、佇列和主題的許可會直接在 RabbitMQ 中設定。這是預設方法。如需詳細資訊，請參閱簡易身分驗證和授權。

OAuth 2.0 身分驗證和授權

在此方法中，代理程式使用者及其許可由外部 OAuth 2.0 身分提供者 (IdP) 管理。vhost、交換、佇列和主題的使用者身分驗證和資源許可是透過 OAuth 2.0 供應商的範圍系統集中。這可簡化使用者管理，並啟用與現有身分系統的整合。如需詳細資訊，請參閱 OAuth 2.0 身分驗證和授權。

IAM 身分驗證和授權

在此方法中，代理程式使用者透過 AWS IAM 傳出聯合使用 IAM 憑證進行身分驗證。IAM 登入資料用於從 AWS Security Token Service (STS) 取得 JWT 權杖，而這些 JWT 權杖可作為 OAuth 2.0 權杖進行身分驗證。此方法利用 Amazon MQ for RabbitMQ 中現有的 OAuth 2.0 支援，其中 AWS 做為 OAuth 2.0 身分提供者。使用者身分驗證由 AWS IAM 處理，而 vhost、交換、佇列和主題的資源許可則透過 RabbitMQ 中設定的 IAM 政策和範圍別名進行管理。如需詳細資訊，請參閱 IAM 身分驗證和授權。

LDAP 身分驗證和授權

在此方法中，代理程式使用者及其許可由外部 LDAP 目錄服務管理。使用者身分驗證和資源許可是透過 LDAP 伺服器集中，允許使用者使用其現有的目錄服務憑證來存取 RabbitMQ。如需詳細資訊，請參閱 LDAP 身分驗證和授權。

HTTP 身分驗證和授權

在此方法中，代理程式使用者及其許可由外部 HTTP 伺服器管理。使用者身分驗證和資源許可是透過 HTTP 伺服器集中，允許使用者使用自己的身分驗證和授權提供者來存取 RabbitMQ。如需此方法的詳細資訊，請參閱 HTTP 身分驗證和授權。

SSL 憑證身分驗證

Amazon MQ 支援 RabbitMQ 代理程式的交互 TLS (mTLS)。SSL 身分驗證外掛程式使用來自 mTLS 連線的用戶端憑證來驗證使用者。在此方法中，代理程式使用者會使用 X.509 用戶端憑證進行身分驗證，而不是使用者名稱和密碼憑證。用戶端的憑證會根據信任的憑證授權單位 (CA) 進行驗證，使用者名稱會從憑證中的欄位擷取，例如通用名稱 (CN) 或主體別名 (SAN)。此方法提供強式身分驗證，無需透過網路傳輸登入資料。如需詳細資訊，請參閱 SSL 憑證身分驗證。

> **Note**
>
> RabbitMQ 支援多個同時使用的身分驗證和授權方法。例如，您可以同時啟用 OAuth 2.0 和簡單（內部）身分驗證。如需詳細資訊，請參閱啟用 OAuth 2.0 和簡單（內部）身分驗證的 OAuth 2.0 教學課程一節，以及 RabbitMQ 存取控制文件。 OAuth

Amazon MQ for ActiveMQ 的身分驗證和授權

Amazon MQ for ActiveMQ 支援下列身分驗證和授權方法：

簡單身分驗證和授權

在此方法中，代理程式使用者是透過 Amazon MQ 主控台或 API 建立和管理。使用者可以設定特定許可來存取佇列、主題和 ActiveMQ Web 主控台。如需此方法的詳細資訊，請參閱建立 ActiveMQ 代理程式使用者。

LDAP 身分驗證和授權

在此方法中，代理程式使用者透過儲存在 LDAP 伺服器的登入資料進行身分驗證。您可以新增、刪除和修改使用者，並透過 LDAP 伺服器將許可指派給主題和佇列，提供集中式身分驗證和授權。如需此方法的詳細資訊，請參閱將 ActiveMQ 代理程式與 LDAP 整合。

升級 Amazon MQ 代理程式引擎版本

Amazon MQ 會定期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。新的引擎版本包括安全性修補程式、錯誤修正和其他代理程式引擎改進。

Amazon MQ 根據語意版本控制規格將版本編號組織為 `X.Y.Z`。在 Amazon MQ 實作中， `X`表示主要版本， `Y`表示次要版本， `Z`表示修補程式版本編號。Amazon MQ 支援兩種類型的升級：

- 主要版本升級 – 發生於主要引擎版本號碼變更時。例如，從 RabbitMQ 3.13 版升級至 4.2 版視為主要版本升級。
- 次要版本升級 – 僅發生於次要引擎版本號碼變更時。例如，從 3.11 版升級至 3.12 版會被視為次要版本升級。

您可以隨時手動將代理程式升級至下一個支援的主要或次要版本。在排程維護時段期間，Amazon MQ 會管理所有代理程式升級至最新支援的修補程式版本。手動和自動版本升級都會在排定的維護時段期間或重新啟動代理程式之後進行。當目前的次要版本終止支援時，Amazon MQ 會將您的代理程式升級至下一個次要版本。

手動升級引擎版本

您可以使用 AWS CLI、 AWS 管理主控台或 Amazon MQ API 來升級代理程式的引擎版本。

AWS 管理主控台

使用 升級代理程式的引擎版本 AWS 管理主控台

1. 在代理程式詳細資訊頁面上，選擇 Edit (編輯)。

2. 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。
3. 捲動到頁面底部，然後選擇 Schedule modification (排程修改)。
4. 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**
   >
   > 單一執行個體代理程式在重新啟動時處於離線狀態。對於叢集代理程式，代理程式重新啟動時一次只會關閉一個節點。

5. 選擇 Apply (套用) 以完成變更套用。

AWS CLI

使用 升級代理程式的引擎版本 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`。
   - `--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）如果您想要立即升級引擎版本，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**
>
> 單一執行個體代理程式在重新啟動時處於離線狀態。對於叢集代理程式，代理程式重新啟動時一次只會關閉一個節點。

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）如果您想要立即升級引擎版本，請使用 RebootBroker API 操作重新啟動代理程式。`broker-id` 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**
>
> 單一執行個體代理程式在重新啟動時處於離線狀態。對於叢集代理程式，代理程式重新啟動時一次只會關閉一個節點。

升級 Amazon MQ 代理程式執行個體類型

> **Important**
>
> `mq.m7g.x` 執行個體僅適用於 Amazon MQ for RabbitMQ 代理程式。Amazon MQ for ActiveMQ 代理程式僅使用`mq.m5.x`執行個體。

中介裝置執行個體類別 (`m7g`) 和大小 (`large`) 的合併描述稱為中介裝置執行個體類型 （ 例如 `mq.m7g.large`)。選擇執行個體類型時，請務必考量會影響代理程式效能的因素：

- 用戶端和佇列的數量
- 傳送的訊息量
- 訊息保留在記憶體中
- 備援訊息

建議僅將較小的中介裝置執行個體類型 (`mq.m7g.medium`) 用於測試應用程式效能。對於用戶端和佇列的生產層級、高輸送量、記憶體中的訊息和備援訊息，我們建議使用較大的中介裝置執行個體類型 (`mq.m7g.large` 和更高版本）。

如果您遇到效能問題，或是要從測試轉移到生產環境，建議您升級至較大的執行個體類型 （ 即從 `micro` 升級至 `large`)。若要升級執行個體類型，您可以使用 AWS 管理主控台 AWS CLI、 或 Amazon MQ API。

AWS 管理主控台

若要使用 升級到較大的執行個體類型 AWS 管理主控台，請執行下列動作：

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. 在規格下，針對中介裝置執行個體類型，從下拉式清單中選擇新的執行個體類型。
5. 在頁面底部，選擇排程修改。
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。

   - 選擇下次重新開機後，如果您希望 Amazon MQ 在下一個排定的維護時段完成升級。
   - 如果您想要重新啟動代理程式並立即升級執行個體類型，請選擇立即。

   > **Important**
   >
   > 重新啟動時，單一執行個體代理程式處於離線狀態。對於叢集代理程式，代理程式重新啟動時一次只會關閉一個節點。

7. 選擇 Apply (套用) 以完成變更套用。

AWS CLI

使用 升級代理程式的執行個體類型 AWS CLI

1. 使用 modify-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。
   - `--host-instance-type` – 要升級至的代理程式引擎版本號碼。

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

2. （選用）如果您想要立即升級執行個體類型，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**
>
> 重新啟動時，單一執行個體代理程式處於離線狀態。對於叢集代理程式，代理程式重新啟動時一次只會關閉一個節點。

Amazon MQ API

使用 Amazon MQ API 升級代理程式的執行個體類型

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《》中的 Amazon MQ 端點和配額AWS 一般參考。

   ```
   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`host-instance-type`中使用 來指定要升級的代理程式執行個體類型。

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

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**
>
> 重新啟動時，單一執行個體代理程式處於離線狀態。對於叢集代理程式，代理程式重新啟動時一次只會關閉一個節點。

Amazon MQ for ActiveMQ 儲存類型

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 • 位置資料更新 • 經常變更的資料	5,000	無

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
Amazon EBS	持續	• 大量文字 • 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。

設定私有 Amazon MQ 代理程式

私有代理程式沒有公有可存取性，也無法從 VPC 外部存取。在設定私有代理程式之前，請檢視下列有關 VPCs、子網路和安全群組的資訊：

- VPC
  - 代理程式的子網路 (s) 和安全群組必須位於相同的 VPC 中。
  - 當您使用私有代理程式時，可能會看到您未使用 VPC 設定的 IP 地址。這些是來自 Amazon MQ 基礎設施的 IP 地址，而且不需要任何動作。
- 子網路
  - 如果子網路位於共用 VPC 內，VPC 必須由建立代理程式的相同帳戶擁有。
  - 如果未提供子網路，則會使用預設 VPC 中的預設子網路。
  - 建立代理程式後，便無法變更使用的子網路。
  - 對於叢集和作用中/待命代理程式，子網路必須位於不同的可用區域。
  - 對於單一執行個體代理程式，您可以指定要使用的子網路，並在相同的可用區域內建立代理程式。
- 安全群組

- 如果未提供安全群組，則會使用預設 VPC 中的預設安全群組。
- 單一執行個體、叢集和作用中/待命代理程式至少需要一個安全群組 （例如預設安全群組）。

> **Note**
>
> 公有 RabbitMQ 代理程式不使用子網路或安全群組。

- 建立代理程式後，便無法變更所使用的安全群組。安全群組本身仍可修改。

在 中設定私有代理程式 AWS 管理主控台

若要設定私有代理程式，請在 中開始建立新的代理程式 AWS 管理主控台。然後，在網路設定區段中，若要設定代理程式的連線，請執行下列動作：

1. 選擇代理程式的私有存取。若要連線至私有代理程式，您可以使用 IPv4, IPv6 或雙堆疊 (IPv4 和 IPv6)。如需詳細資訊，請參閱Connecting to Amazon MQ。
2. 接著，選擇使用預設 VPC、子網路 (s) 和安全群組 (s)，或選擇選取現有的 VPC、子網路 (s) 和安全群組 (s)。如果您不想使用預設或現有的 VPC、子網路 （子網路） 或安全群組 （安全群組），您必須建立新的 VPC 或安全群組，才能連線至私有代理程式。

> **Note**
>
> 對於私有代理程式存取，連線方法將與子網路的所選 IP 類型相同。建立代理程式後，就無法變更 VPC 端點，且一律會有所選子網路的 IP 類型。如果您想要使用新的 IP 類型，則必須建立新的代理程式。

> **Note**
>
> Amazon MQ for ActiveMQ 不使用 VPC 端點。當您首次建立 ActiveMQ 代理程式時，Amazon MQ 會在 VPC 中佈建彈性網路界面 (ENI)。安全群組會放置在 ENI 中，可用於公有和私有代理程式。

在沒有公開存取性的情況下存取 Amazon MQ 代理程式 Web 主控台

當您關閉代理程式的公有可存取性時，建立代理程式 AWS 的帳戶 ID 可以存取私有代理程式。如果您關閉代理程式的公有可存取性，則必須執行下列步驟來存取代理程式 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。

> **Important**
>
> 如果您使用私有代理程式，可能會看到您未使用 VPC 設定的 IP 地址。這些是來自 Amazon MQ 基礎設施上 RabbitMQ 的 IP 地址，而且不需要任何動作。 Amazon MQ

排程 Amazon MQ 代理程式的維護時段

Amazon MQ 會在維護時段期間定期對訊息中介裝置的硬體、作業系統或引擎軟體執行維護。例如，如果您變更代理程式執行個體類型，Amazon MQ 會在下一個排定的維護時段套用您的變更。視您訊息代理程式排定的操作而定，維護的持續時間最多可持續兩小時。您可以透過選取跨多個可用區域 (AZ) 高可用性的代理程式部署模式，將維護時段期間的停機時間降至最低。

Amazon MQ for ActiveMQ 提供主動/待命部署，以實現高可用性。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，且至少有一個執行個體仍然可用。此外，您可以設定代理程式網路，維護時段在一週內會有所不同。Amazon MQ for RabbitMQ 提供高可用性的叢集部署。在叢集部署中，Amazon MQ 透過隨時保持至少兩個執行中的節點，一次執行一個節點的維護操作。

第一次建立代理程式時，您可以將維護時段排定在指定時間每週執行一次。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。代理程式維護時段完成後，Amazon MQ 會重設限制，您可以在下一個維護時段發生之前再次調整排程。調整代理程式維護時段時，代理程式可用性不會受到影響。

若要調整代理程式維護時段，您可以使用 AWS 管理主控台 AWS CLI、 或 Amazon MQ API。

使用 排程代理程式維護時段 AWS 管理主控台

使用 調整中介裝置維護時段 AWS 管理主控台

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. 接著，選取排程修改。然後選擇下次重新開機後或立即。選擇下次重新啟動後， 會立即更新維護時段，而不會重新啟動代理程式。選擇立即將立即重新啟動代理程式。
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。

使用 排程代理程式維護時段 AWS CLI

使用 調整中介裝置維護時段 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
   ```

使用 Amazon MQ API 排程代理程式維護時段

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《》中的 Amazon MQ 端點和配額AWS 一般參考。

   ```
   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. (選用) 使用 DescribeBroker API 操作，以確認已成功更新維護時段。broker-id 被指定為路徑參數。

```
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
```

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 管理主控台來重新啟動 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

如果您不使用 Amazon MQ 代理程式（且近期內未預見使用該代理程式），最佳實務是從 Amazon MQ 將其刪除，以降低成本 AWS 。

以下範例示範如何使用 AWS 管理主控台來刪除代理程式。

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。

Amazon MQ 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	Description
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。

主控台	API	Description
正在重新啟動	REBOOT_IN_PROGRESS	目前正在重新啟動代理程式。
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 疑難排解 中選擇需採取動作的代碼，以找到解決問題的說明。

將標籤新增至 Amazon MQ 資源

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組織 AWS 帳單，以反映您自己的成本結構。若要這樣做，請註冊以取得 AWS 您的帳戶帳單，以包含標籤索引鍵和值。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	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 主控台中新增標籤

您可以依照下列步驟，快速將標籤新增至您在 Amazon MQ 主控台中建立的資源：

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)。

新增標籤後，您可以在 Amazon MQ 主控台中檢視、編輯和移除資源的標籤。您也可以使用 REST API 檢視資源的標籤。如需詳細資訊，請參閱 Amazon MQ REST API 參考。

使用 Amazon MQ for 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 、Amazon MQ REST API 或 建立 AWS 管理主控台、管理和刪除代理程式 AWS Command Line Interface。

Amazon MQ for ActiveMQ 代理程式可以部署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

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

Amazon MQ for ActiveMQ 代理程式

什麼是 Amazon MQ for 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。中介裝置執行個體類別 (m5) 和大小 (`large`、`medium`) 的合併描述稱為中介裝置執行個體類型 （ 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 Amazon MQ for ActiveMQ 代理程式的部署選項。

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：

- 入門：建立並連線至 ActiveMQ 代理程式
- 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

Attributes

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 ActiveMQ 代理程式使用者
- 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

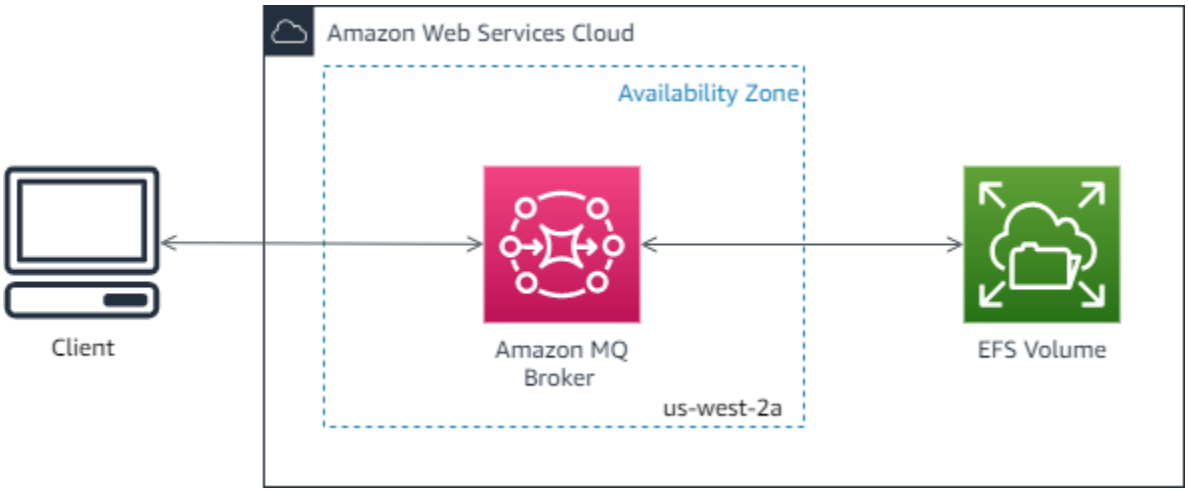
Amazon MQ for ActiveMQ 代理程式的部署選項

Amazon MQ 為代理程式提供單一執行個體和叢集部署選項。

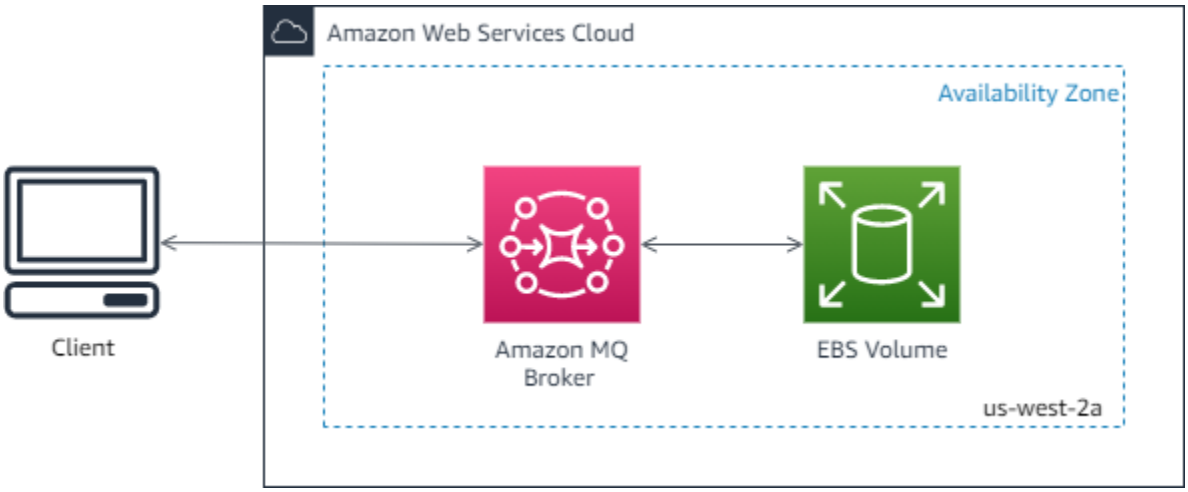
選項 1：Amazon MQ 單一執行個體代理程式

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。



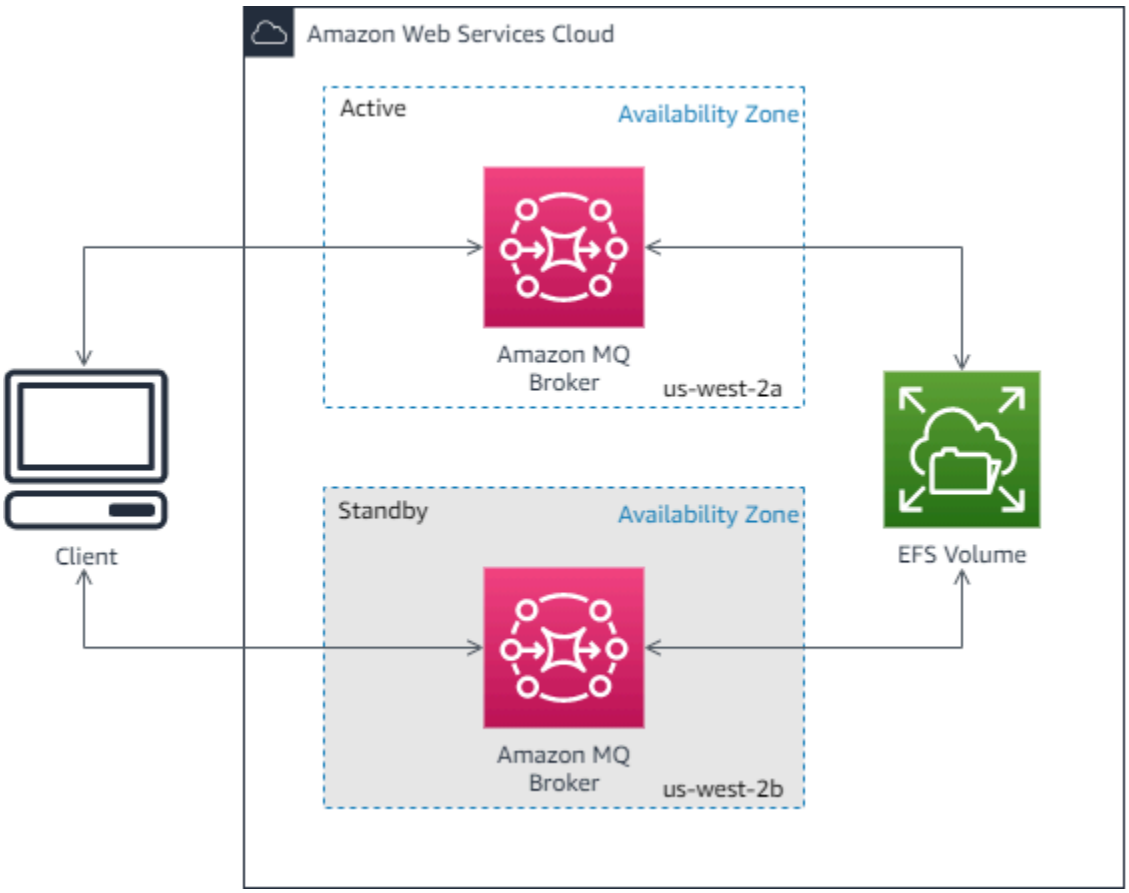
選項 2：Amazon MQ 作用中/待命代理程式可提供高可用性

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。您初始化的維護時段和代理程式重新啟動會導致容錯移轉發生。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。對於線路層級通訊協定端點，您應該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。



代理程式的 Amazon MQ 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。建立中介裝置網路可以提高可用性、容錯能力，以及與多個中介裝置執行個體進行負載平衡。

中介裝置網路的運作方式為何？

透過使用網路連接器將一個中介裝置連接到另一個中介裝置來建立中介裝置網路。網路連接器提供從一個代理程式到另一個代理程式的隨需訊息。網路連接器在代理程式組態中設定為非雙工或雙工連線。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。對於雙工連線，訊息會在兩個代理程式之間雙向轉送。

如果網路連接器設定為雙工，訊息也會從 Broker2 轉送到 Broker1。

您可以在中介裝置網路中使用非雙工和雙工連線。您可能想要將雙工連線引入另一個代理程式，以改善流量，或避免提高限制。雙工連線也適用於從內部部署到 Amazon MQ 受管代理程式的部分遷移。

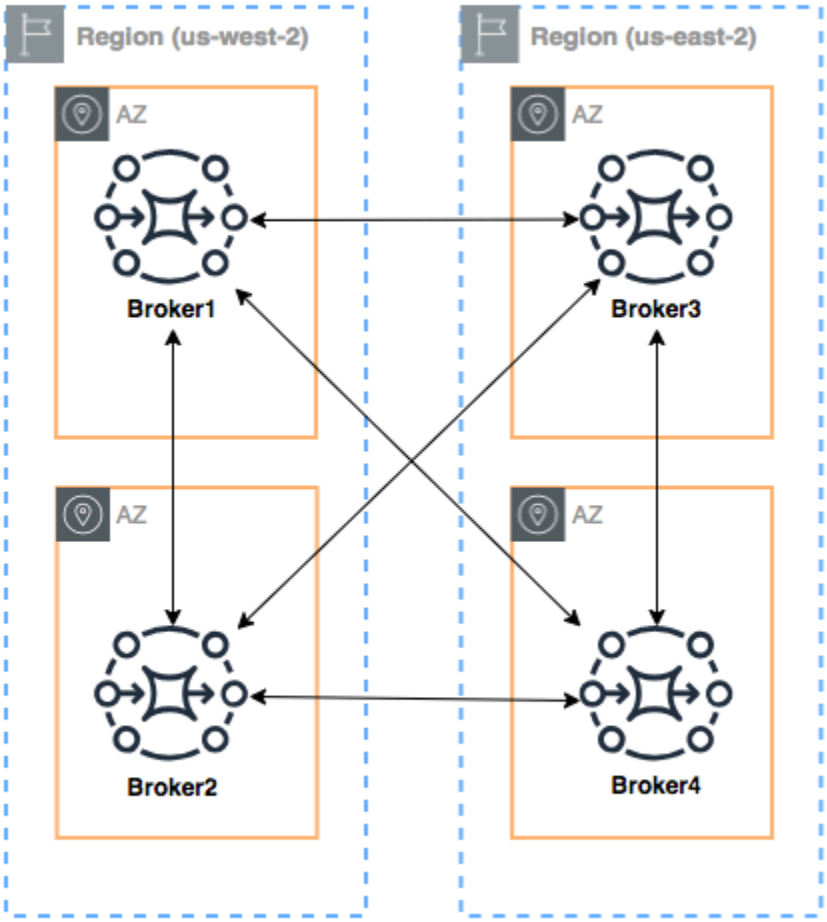
代理程式網路如何處理登入資料？

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。如需詳細資訊，請參閱Configure Network Connectors for Your Broker。

跨區域

若要設定跨越 AWS 區域的代理程式網路，請在這些區域中部署代理程式，並將網路連接器設定為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-
west-2.amazonaws.com:61617)"/>
    <networkConnector name="1_to_3" userName="myCommonUser" duplex="true"
        uri="static:(ssl://b-743c885d-2244-4c95-af67-a85017ff234e-3.mq.us-
east-2.amazonaws.com:61617)"/>
    <networkConnector name="1_to_4" userName="myCommonUser" duplex="true"
        uri="static:(ssl://b-62a7fb31-d51c-466a-a873-905cd660b553-4.mq.us-
east-2.amazonaws.com:61617)"/>
```

```
</networkConnectors>
```

Broker2 的網路連接器：

```
<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-
west-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 代理程式執行個體類型

中介裝置執行個體類別 (`m5`) 和大小 (`large`、`medium`) 的合併描述稱為中介裝置執行個體類型 （ 例如 `mq.m5.large`)。下表列出 Amazon MQ ActiveMQ 代理程式執行個體類型。

Amazon MQ 至少會在執行個體類型終止支援前 90 天發出通知。我們建議您在end-of-support日期之前將代理程式升級至新的執行個體類型，以防止任何中斷。

> **Important**
>
> 您無法在 2025 年 3 月 17 日`mq.m4.large`當天`t2.micro`或之後建立代理程式。

執行個體類型	vCPU	記憶體 (GiB)	建議用途	儲存	Amazon MQ 上的終止支援
`mq.t3.micro`	2	1	評估	EFS	
`mq.m5.large`	2	8	生產	EFS 或 EBS	
`mq.m5.xlarge`	4	16	生產	EFS 或 EBS	

執行個體類型	vCPU	記憶體 (GiB)	建議用途	儲存	Amazon MQ 上的終止支援
mq.m5.2xlarge	8	32	生產	EFS 或 EBS	
mq.m5.4xlarge	16	64	生產	EFS 或 EBS	

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 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。

> **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。
>
> 您只能使用 `DeleteConfiguration` API 刪除組態。如需詳細資訊，請參閱《Amazon MQ API 參考》中的組態。

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

您可以使用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 Amazon MQ for 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。

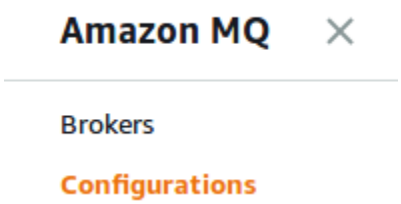
以下範例顯示如何使用 AWS 管理主控台建立和套用 Amazon MQ 代理程式組態。

> **Important**
>
> 您只能使用 `DeleteConfiguration` API 刪除組態。如需詳細資訊，請參閱《Amazon MQ API 參考》中的組態。

建立新組態

若要建立新的代理程式組態，請先建立新的組態。

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. 選擇建立組態。

建立新的組態修訂

建立代理程式組態之後，您將需要使用組態修訂來編輯組態。

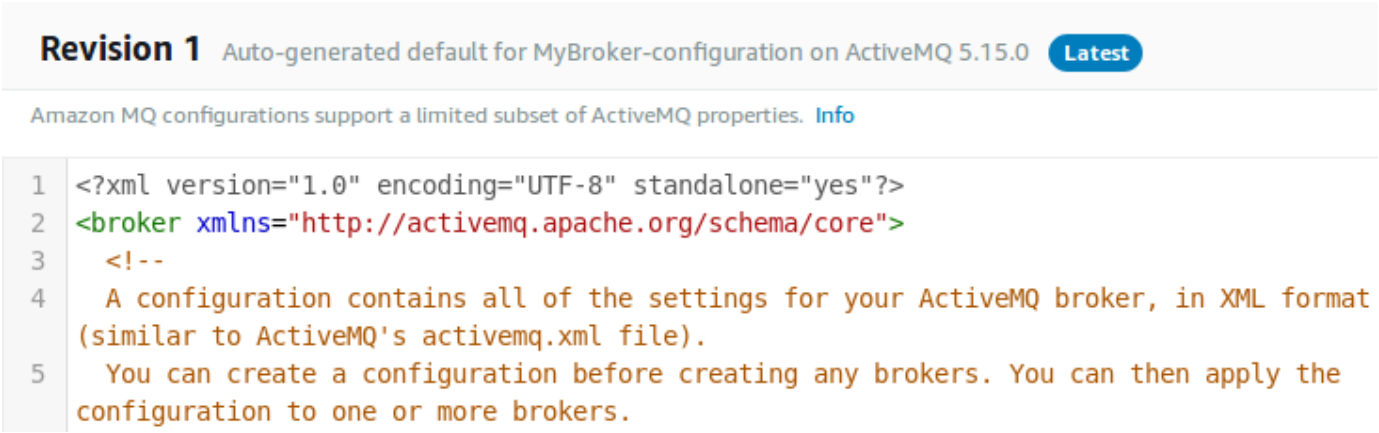
1. 從組態清單中，選擇 ***MyConfiguration***。

> **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，會顯示新組態修訂使用的代理程式引擎類型和版本 （ 例如 Apache ActiveMQ 5.15.16)。

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)。

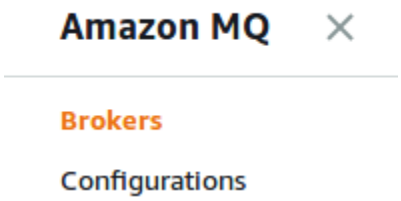
   隨即儲存組態的新修訂版。

> **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。

將組態修訂套用至您的代理程式

修改組態之後，您可以將組態修訂套用至代理程式。

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**
>
> 重新啟動時，單一執行個體代理程式處於離線狀態。對於叢集代理程式，代理程式重新啟動時一次只會關閉一個節點。

5. 選擇套用。

   組態修訂會在指定時間套用至代理程式。

編輯 Amazon MQ for ActiveMQ 組態修訂

在將組態修訂套用至代理程式之後，您可能想要編輯組態修訂。使用下列指示來編輯組態修訂。

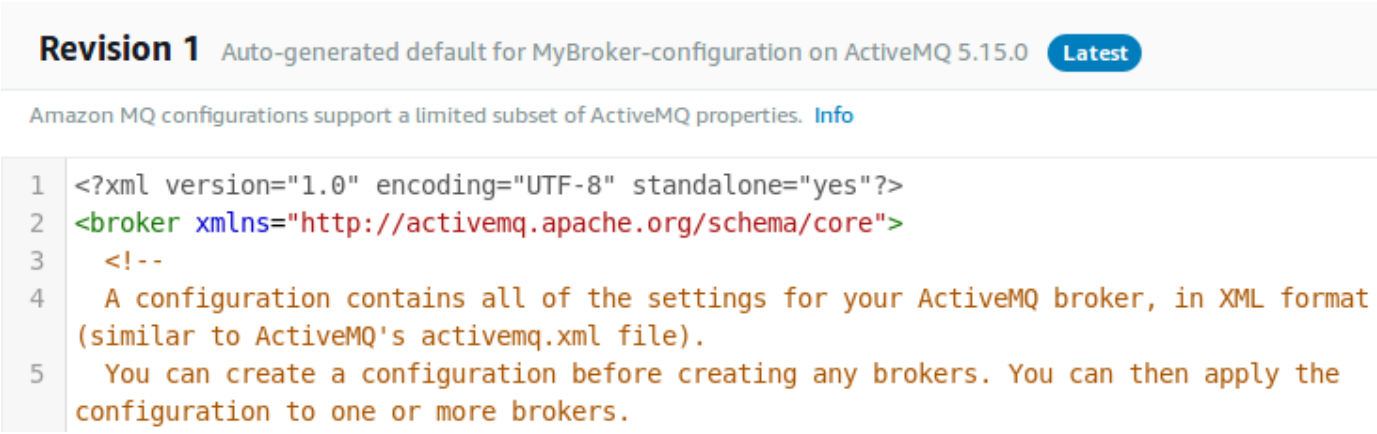
1. 登入 Amazon MQ 主控台。
2. 從代理程式清單，選取您的代理程式 (例如 MyBroker)，然後選擇 Edit (編輯)。
3. 在 ***MyBroker*** 頁面上，選取 Edit (編輯)。
4. 在 Edit ***MyBroker*** (編輯 MyBroker) 頁面的 Configuration (組態) 區段中，選取 Configuration (組態) 和 Revision (修訂)，然後選擇 Edit (編輯)。

> **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。

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。

Element
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)
discardingDLQBrokerPlugin (屬性)
fileCursor
fileDurableSubscriberCursor

Element
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
queryBasedSubscriptionRecoveryPolicy (屬性)
queue (屬性)
redeliveryPlugin (屬性 | 子集合元素)

Element
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)
transportConnector (屬性)
uniquePropertyMessageEvictionStrategy (屬性)
virtualDestinationInterceptor (子集合元素)

Element
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。

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

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

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

Element	屬性
	useVirtualDestSubsOnCreation
	useVirtualTopics
cachedMessageGroupMapFactory	cacheSize
compositeQueue	concurrentSend
	copyMessage
	forwardOnly
	name
	sendWhenNotMatched
compositeTopic	concurrentSend
	copyMessage
	forwardOnly
	name
	sendWhenNotMatched
conditionalNetworkBridgeFilterFactory	rateDuration
	rateLimit
	replayDelay
	replayWhenNoConsumers
	selectorAware 支援於 Apache ActiveMQ 5.16.x

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

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

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

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

Element	屬性
	reduceMemoryFootprint
	sendAdvisoryIfNoConsumers
	sendFailIfNoSpace
	sendFailIfNoSpaceAfterTimeout 支援於 Apache ActiveMQ 5.16.4 及更新版本
	sendDuplicateFromStoreToDLQ
	storeUsageHighWaterMark
	strictOrderDispatch
	tempQueue
	tempTopic
	timeBeforeDispatchStarts
	topic
	topicPrefetch
	useCache
	useConsumerPriority
usePrefetchExtension	
prefetchRatePendingMessageLimitStrategy	multiplier

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

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

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

Amazon MQ 父元素屬性

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。

屬性

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。

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

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

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

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

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

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

Element	子集合元素	子元素
		lastImageSubscriptionRecoveryPolicy
		noSubscriptionRecoveryPolicy
		queryBasedSubscriptionRecoveryPolicy
		retainedMessageSubscriptionRecoveryPolicy
		timedSubscriptionRecoveryPolicy
sharedDeadLetterStrategy	deadLetterQueue	queue
		tempQueue
		tempTopic
		topic
virtualDestinationInterceptor	virtualDestinations	compositeQueue
		compositeTopic
		virtualTopic

Amazon MQ 子元素屬性

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>
```

> **Note**
>
> Amazon MQ 上 ActiveMQ 中的 `activemq-webconsole`群組具有所有佇列和主題的管理員許可。此群組中的所有使用者都將擁有管理員存取權。

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

使用者名稱

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-west-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>
```

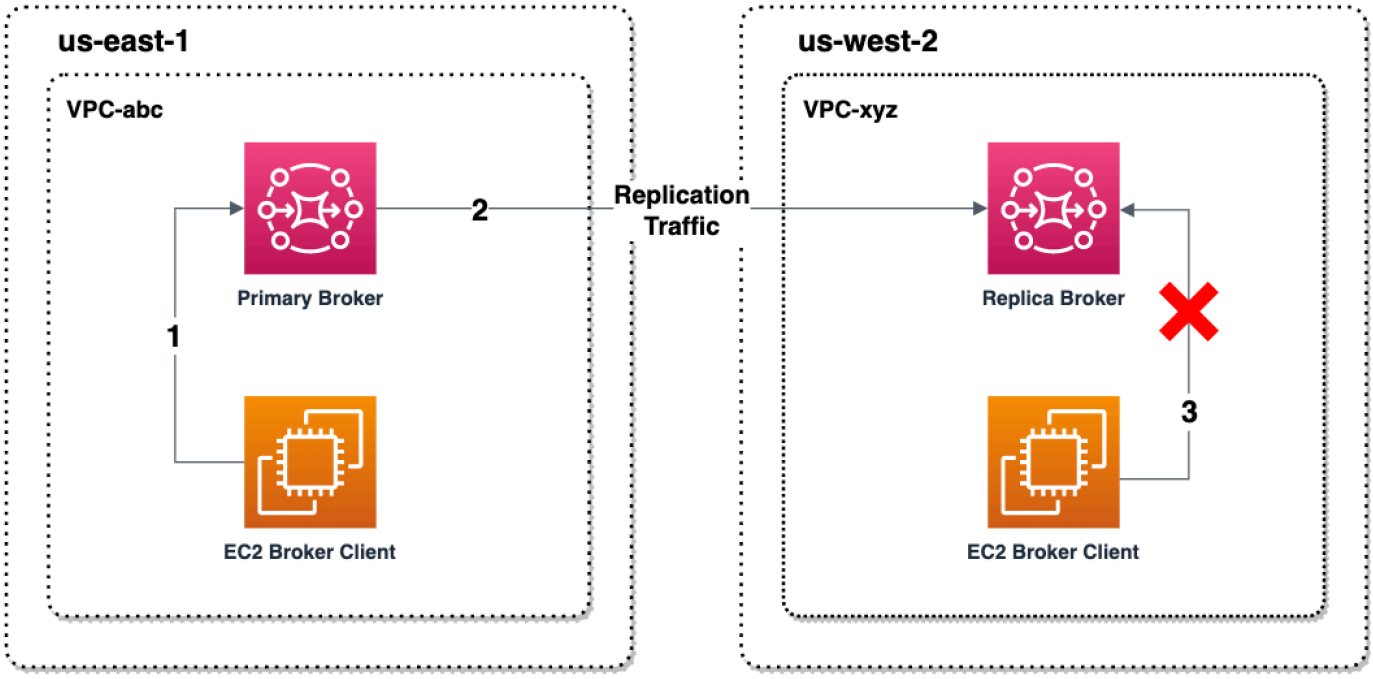
Amazon MQ for ActiveMQ 跨區域資料複寫

Amazon MQ for ActiveMQ 提供跨區域資料複寫 (CRDR) 功能，允許從主要 AWS 區域中的主要代理程式非同步訊息複寫到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。

跨區域資料複寫的主要和複本代理程式

您可以建立主要和複本代理程式，以從主要 AWS 區域中的主要代理程式非同步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 Amazon MQ 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。
> 不適用於鏡像佇列。

建立 Amazon MQ 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 跨區域資料複寫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

先決條件

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。如需詳細資訊，請參閱Amazon MQ for ActiveMQ 代理程式的部署選項。
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。其他 AWS 服務可存取中介裝置名稱，包括 CloudWatch Logs。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。

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。其他 AWS 服務可存取中介裝置使用者名稱，包括 CloudWatch Logs。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。其他 AWS 服務可存取中介裝置使用者名稱，包括 CloudWatch Logs。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。其他 AWS 服務可存取中介裝置使用者名稱，包括 CloudWatch Logs。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 入門：建立並連線至 ActiveMQ 代理程式

刪除 Amazon MQ 跨區域資料複寫代理程式

若要刪除主要或複本跨區域資料複寫 (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 管理主控台取消配對和重新啟動代理程式。

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「確認」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。

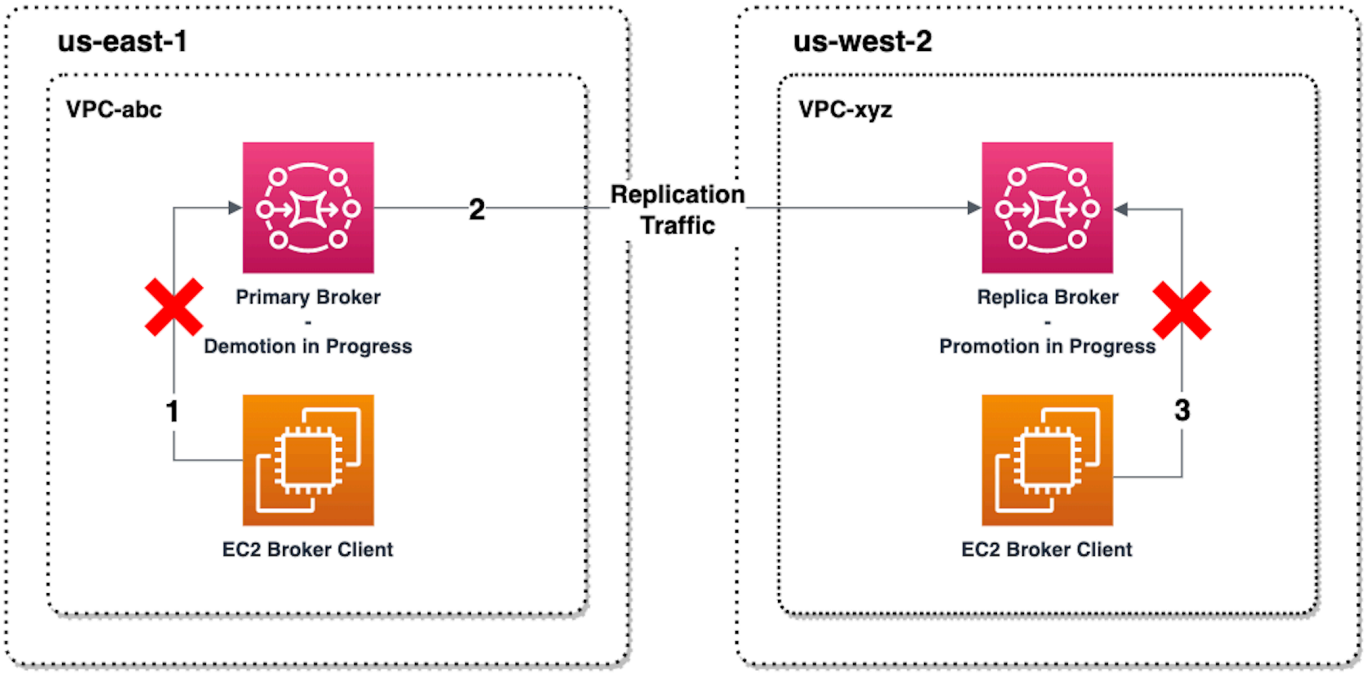
啟動切換或容錯移轉，將 Amazon MQ 複本代理程式提升為主要代理程式角色

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。

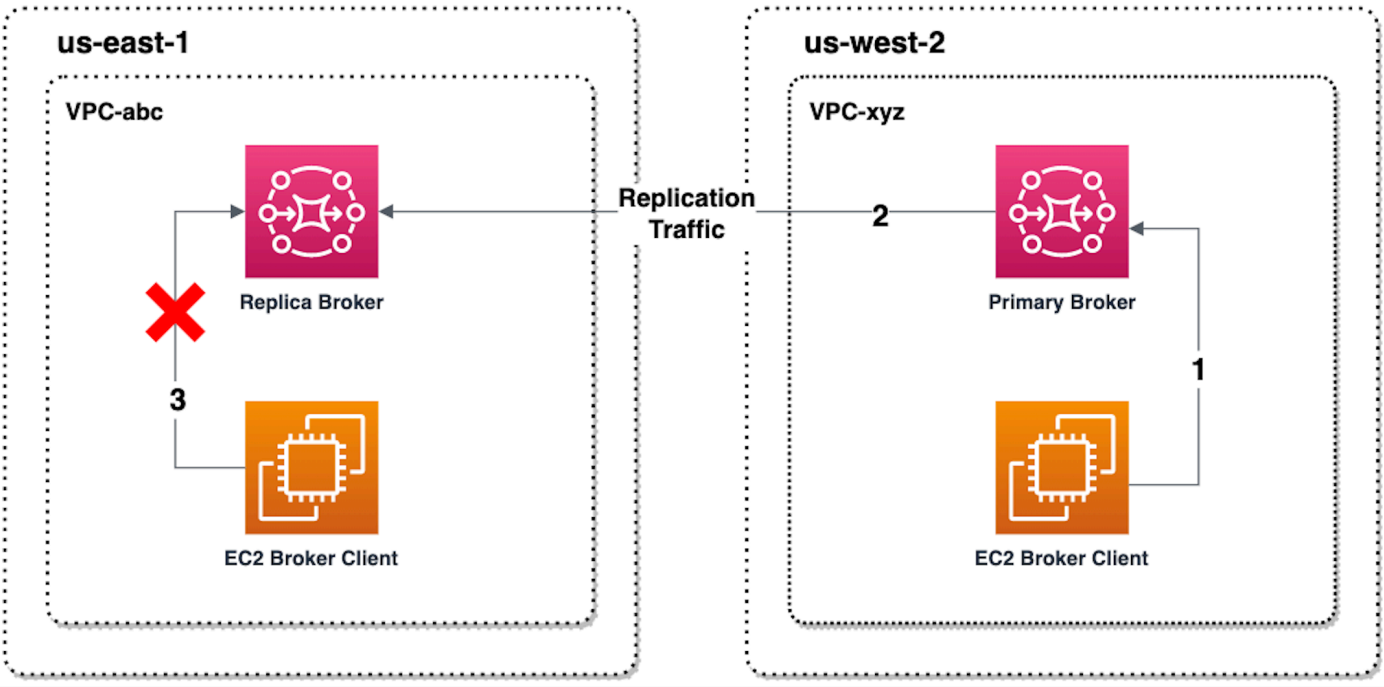
切換會將一致性視為優先於可用性。當此容錯移轉操作完成時，就可保證代理程式擁有一致的狀態。使用切換時，在建立代理程式間一致性的情況下，可能會有一段時間用戶端連線都無法使用代理程式。複本提升時，兩個代理程式會立即擁有相同的狀態。切換是否成功，取決於兩個地區的運作狀態和區域間網路是否成功。

容錯移轉會將可用性視為優先於一致性。當此操作完成時，不保證代理程式具有相同的狀態。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 VPC 中的用戶端無法在操作進行時產生進一步的狀態變更，且主要代理程式正在降級為複本。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。

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 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
- 步驟 3：（選用）連接至 AWS Lambda 函數
- 建立 ActiveMQ 代理程式使用者
- 編輯 ActiveMQ 代理程式使用者
- 刪除 ActiveMQ 代理程式使用者
- 搭配使用 Java Message Service (JMS) 與 ActiveMQ 的運作範例

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。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：

- 代理程式的 Amazon MQ 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，請參閱入門：建立並連線至 ActiveMQ 代理程式。
- 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 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. 針對類型，選取自訂 TCP。

   c. 針對 Port Range (連接埠範圍)，輸入 OpenWire 連接埠 (61617)。

   d. 執行以下任意一項：

   - 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。
      - 執行以下任意一項：
        - 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-
west-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

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

接著，為您的應用程式啟用傳入連線。

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. 針對類型，選取自訂 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>5.15.16</version>
    </dependency>
    <dependency>
        <groupId>org.apache.activemq</groupId>
        <artifactId>activemq-pool</artifactId>
        <version>5.15.16</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

使用下列指示來建立訊息生產者並接收訊息。

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 Amazon MQ for ActiveMQ 代理程式的部署選項。
對於線路層級通訊協定端點，您應該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

使用下列指示來建立訊息生產者並接收訊息。

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 簡訊服務 （ 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**
>
> Amazon MQ 不支援私有 CA 發行的伺服器憑證。

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
- 先決條件
- LDAP 入門
- LDAP 整合的運作方式

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。

> **Important**
>
> 使用 LDAP 時，身分驗證不區分大小寫，但授權對您的使用者名稱區分大小寫。

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)。

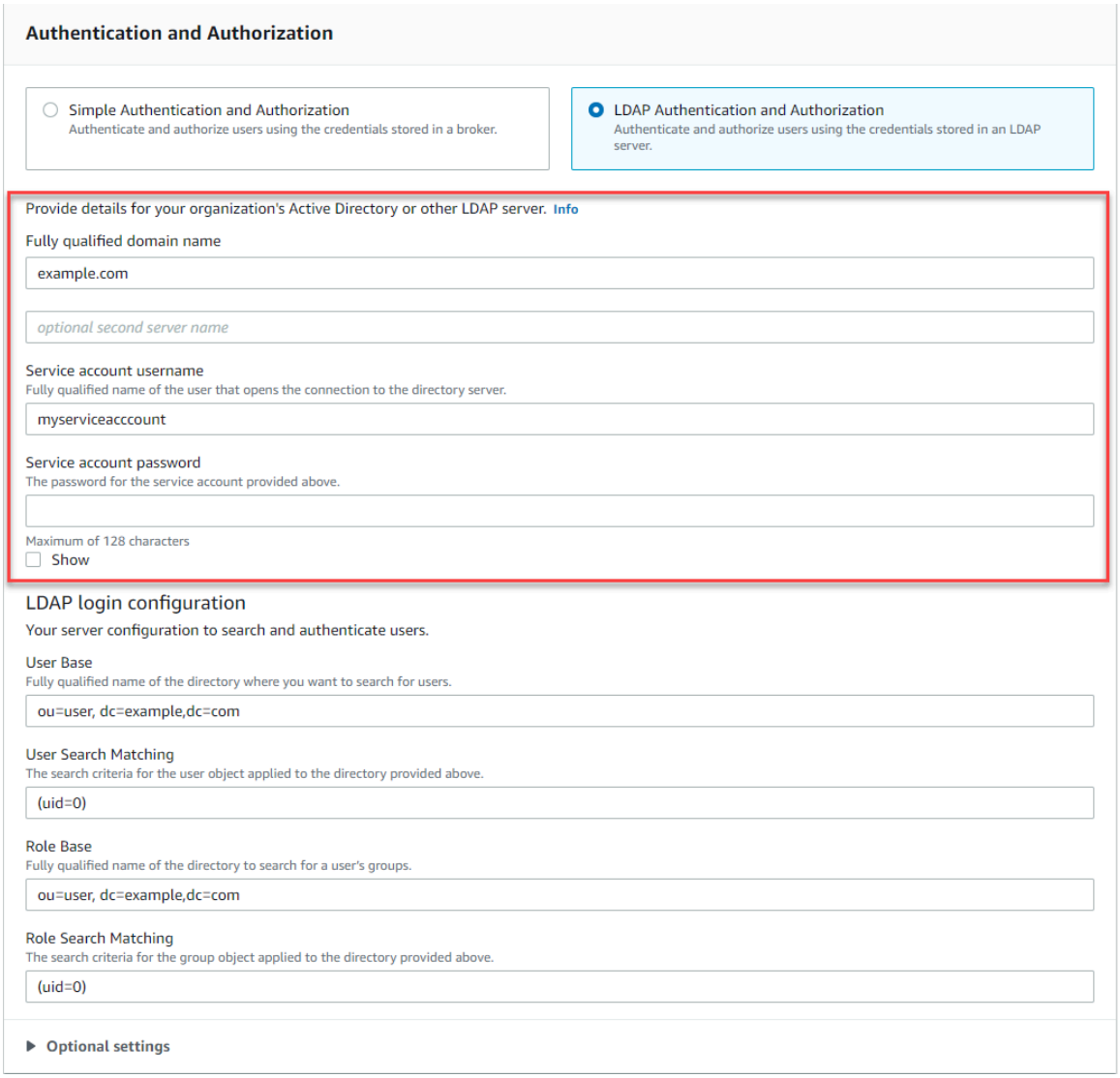
提供服務帳戶的相關資訊：

- 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。

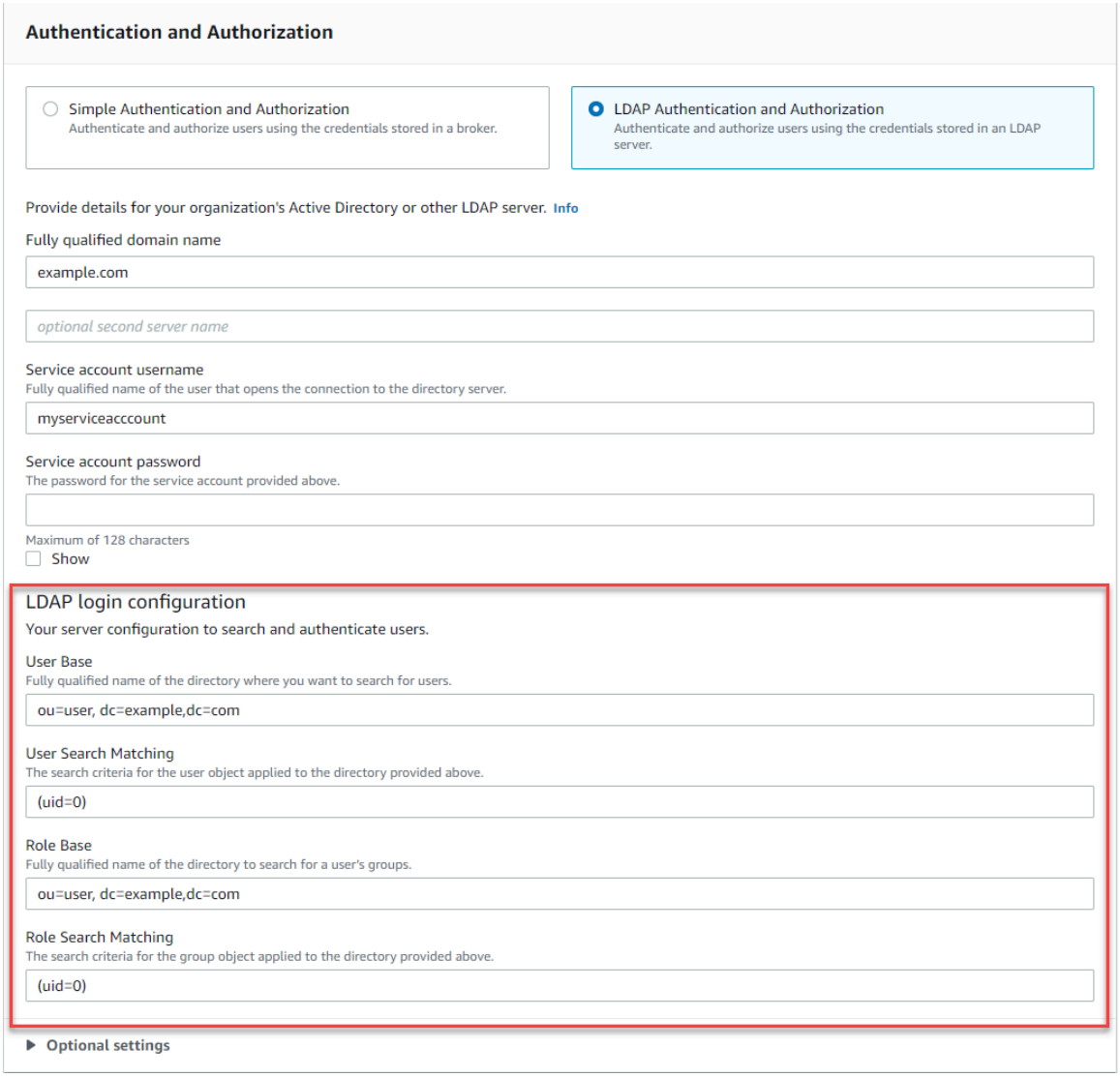


在 LDAP login configuration (LDAP 登入組態) 區段中，提供下列必要資訊：

- 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 Authorization。
- 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：

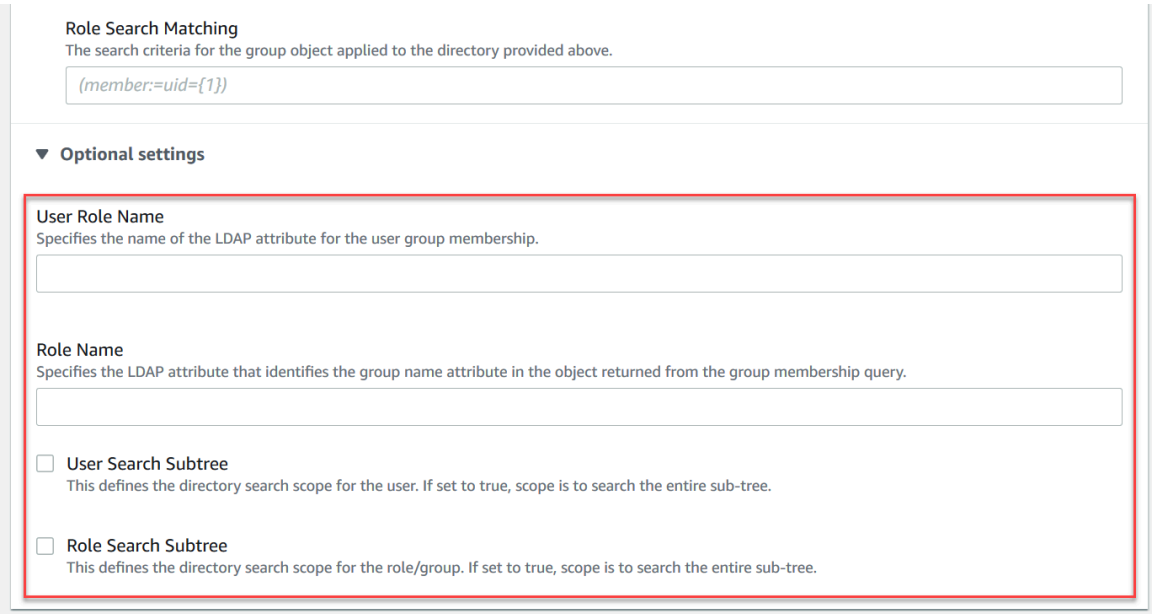
```
dn: uid=jdoe,ou=user,dc=example,dc=com
objectClass: user
uid: jdoe
```

```
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。

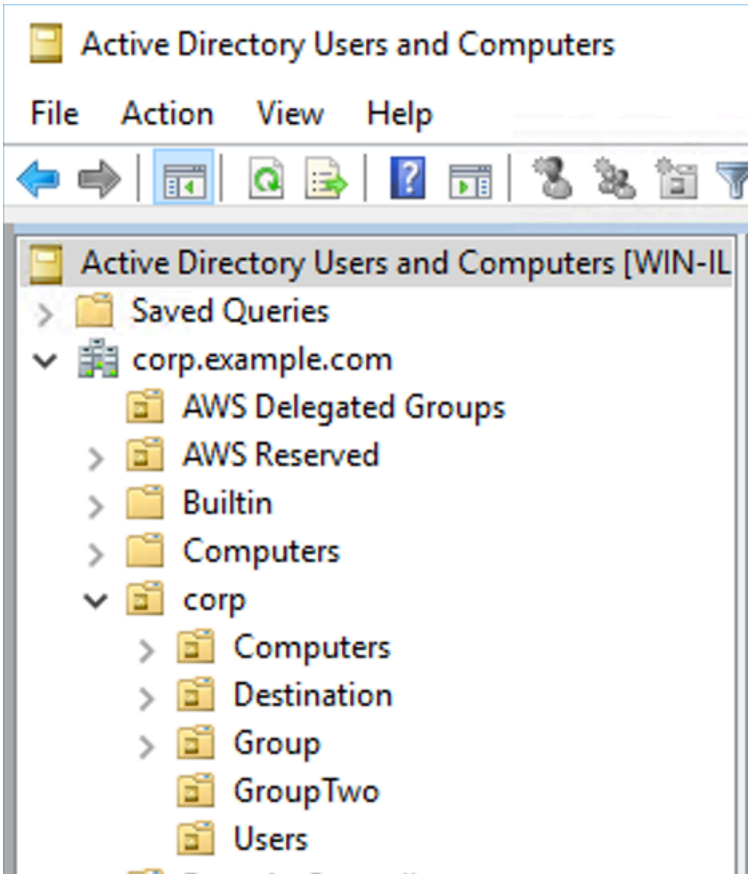
身分驗證

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，如果您使用的是 AWS Managed Microsoft AD，而且您擁有網域元件 `corp`、 `example`和 `com`，並且在您擁有組織單位 `corp`和 的網域元件中`Users`，您會使用下列項目做為您的使用者基礎：

```
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。

Authorization

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**
>
> 若要能夠在代理程式的activemq.xml組態檔案中使用 cachedLDAPAuthorizationMap元素，您必須在透過 建立組態 AWS 管理主控台時選擇 LDAP 身分驗證和授權選項，或透過 設定組態 AWS 管理主控台，或使用 Amazon MQ API 建立新組態LDAP時，將 authenticationStrategy 屬性設定為 。

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 會將您透過 AWS 管理主控台或在 API 請求的 ldapServerMetadata 屬性中提供的值替換為上述屬性。

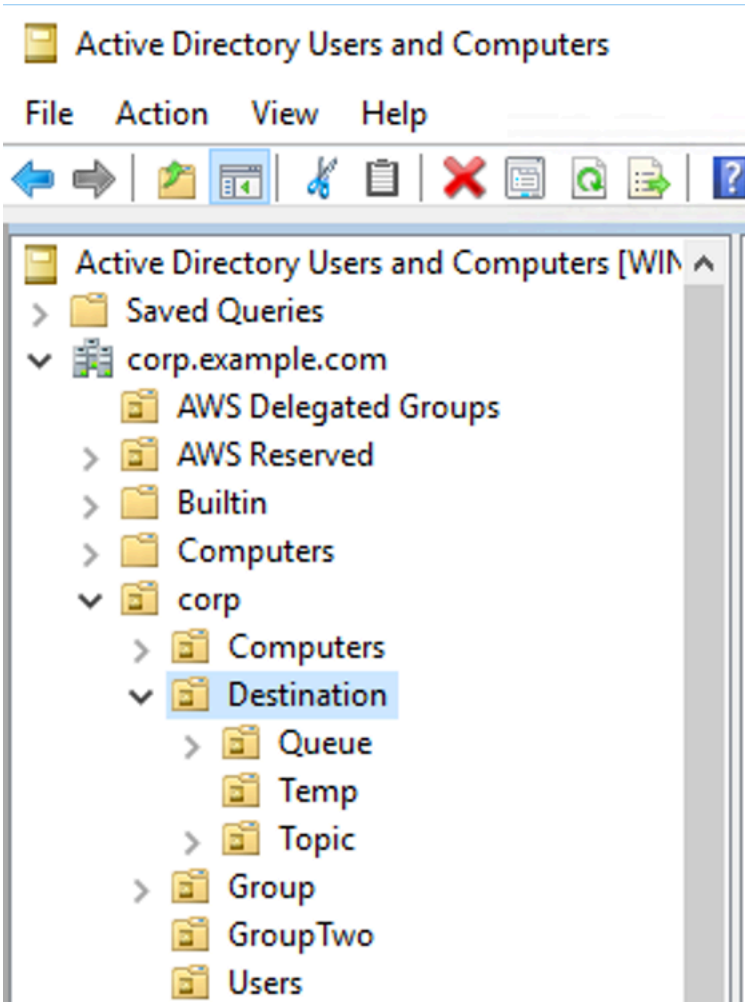
以下示範 cachedLdapAuthorizationMap 的正常運作範例。

```
<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，對於上述 範例 AWS Managed Microsoft AD，使用 `corp`、 `example`和 的相同網域元件`com`，您可以指定名為 的組織單位`destination`，以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：

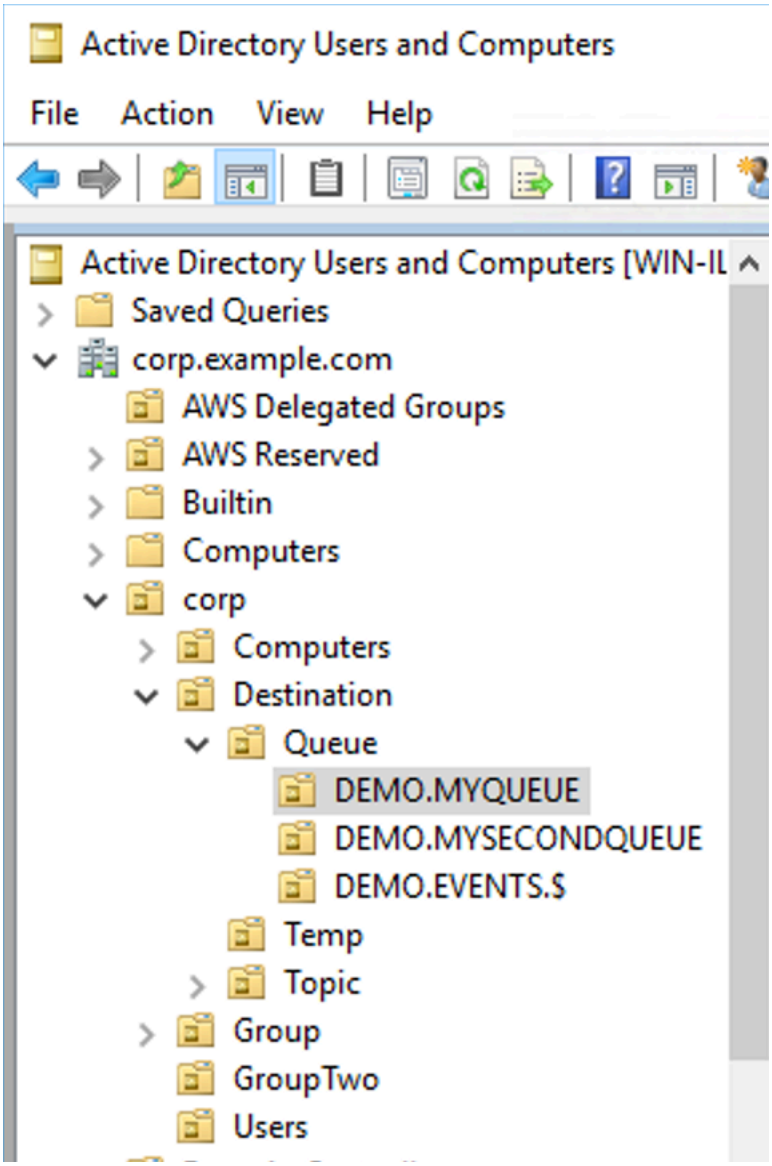
```
OU=DEMO.EVENTS.$,OU=Queue,OU=Destination,OU=corp,DC=corp,DC=example,DC=com
```

> **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：

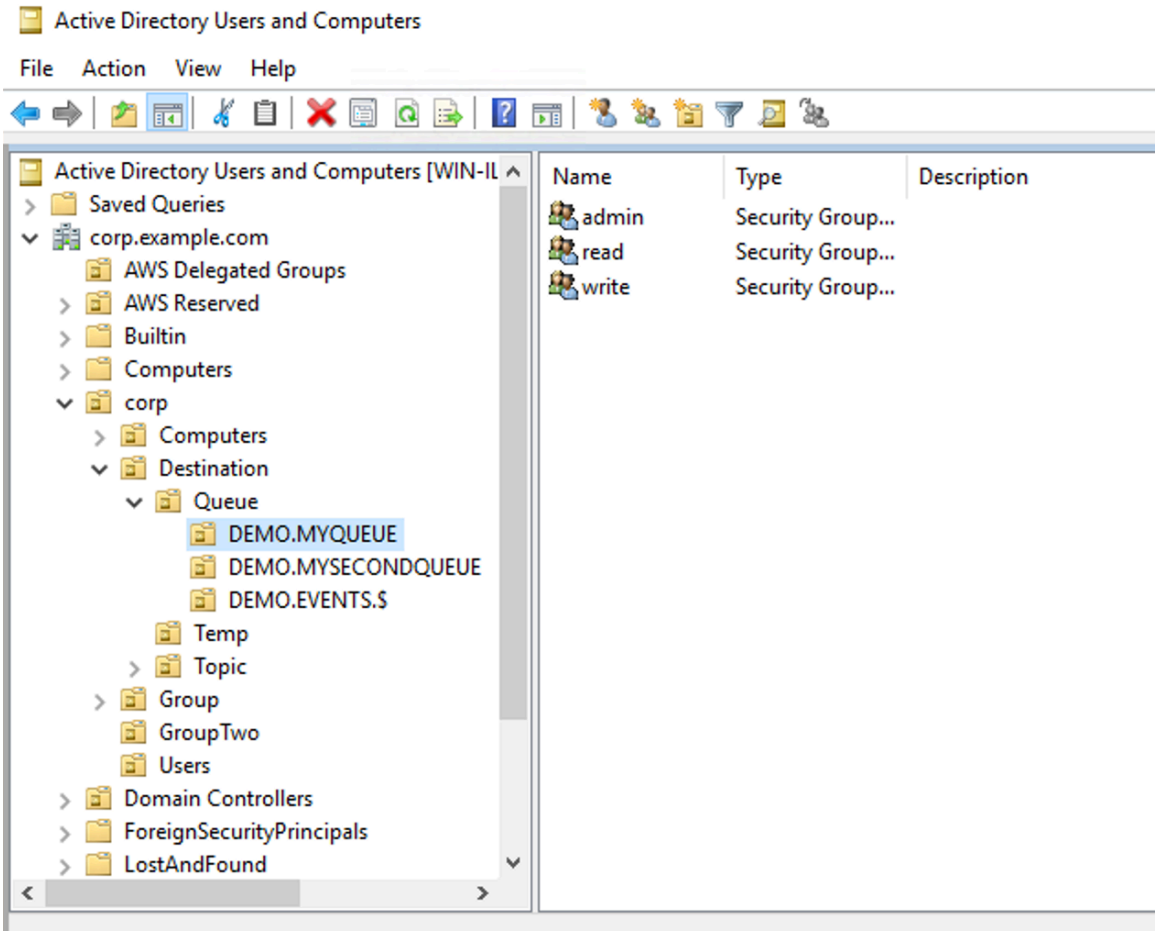
```
OU=DEMO.MYQUEUE,OU=Queue,OU=Destination,OU=corp,DC=corp,DC=example,DC=com
```



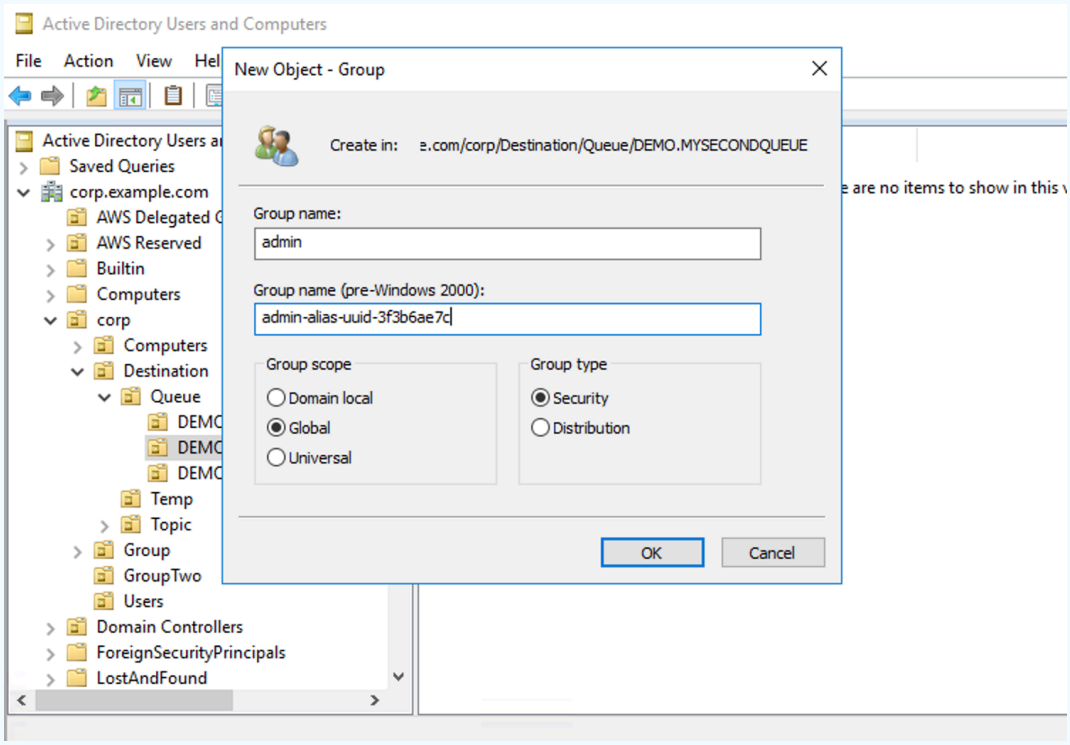
安全群組

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。

步驟 3：（選用）連接至 AWS Lambda 函數

AWS Lambda 可以連線至您的 Amazon MQ 代理程式並使用訊息。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 也必須連線到 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 的介面 VPC 端點。

3. 使用 AWS 管理主控台，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。

建立 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。

> **Note**
>
> Amazon MQ 上 ActiveMQ 中的 `activemq-webconsole`群組具有所有佇列和主題的管理員許可。此群組中的所有使用者都將擁有管理員存取權。

以下範例顯示如何使用 AWS 管理主控台來建立、編輯和刪除 Amazon MQ 代理程式使用者。

建立新的 ActiveMQ 代理程式使用者

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。

編輯 ActiveMQ 代理程式使用者

若要編輯現有的使用者，請執行下列動作：

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。

刪除 ActiveMQ 代理程式使用者

當您不再需要使用者時，您可以刪除該使用者。

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。

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

若要以程式設計方式使用 Amazon MQ，您必須使用傳入連線。

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. 針對類型，選取自訂 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>5.15.16</version>
    </dependency>
    <dependency>
        <groupId>org.apache.activemq</groupId>
        <artifactId>activemq-pool</artifactId>
        <version>5.15.16</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);
                        }
                    }
                    }
```

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 表示主要版本， Y 表示次要版本， Z表示修補程式版本編號。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 5.17 版升級至 6.0 版會被視為主要版本升級。如果只有次要或修補程式版本號碼變更，則版本變更會被視為次要變更。例如，從 5.18 版升級至 5.19 版會被視為次要版本升級。開啟 autoMinorVersionUpgrade 時，Amazon MQ 會將您的代理程式升級至最新的可用修補程式版本。

Amazon MQ for ActiveMQ 建議所有代理程式使用最新支援的次要版本。如需如何升級代理程式引擎版本的指示，請參閱升級 Amazon MQ 代理程式引擎版本。

Amazon MQ for ActiveMQ 支援的引擎版本

Amazon MQ 版本支援行事曆會指出代理程式引擎版本何時會終止支援。當版本終止支援時，Amazon MQ 會自動將此版本的所有代理程式升級至下一個支援的版本。此升級會在代理程式的排程維護時段期間，以及end-of-support日期後的 45 天內進行。

Amazon MQ 至少會在版本終止支援前 90 天發出通知。我們建議您在end-of-support之前升級代理程式，以防止任何中斷。此外，您無法在支援結束日期的 30 天內，在排程終止支援版本上建立新的代理程式。

Apache ActiveMQ 版本	Amazon MQ 上的終止支援
ActiveMQ 5.19（建議）	
ActiveMQ 5.18	
ActiveMQ 5.17	2025 年 6 月 16 日
ActiveMQ 5.16	2024 年 11 月 15 日
ActiveMQ 5.15	2024 年 9 月 16 日

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。如果您在建立代理程式時未指定引擎版本編號，Amazon MQ 會自動預設為最新的引擎版本編號。

引擎版本升級

您可以隨時手動將代理程式升級至下一個支援的主要或次要版本。當您開啟自動次要版本升級時，Amazon MQ 會在維護時段將您的代理程式升級至最新的支援修補程式版本。

如需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`。

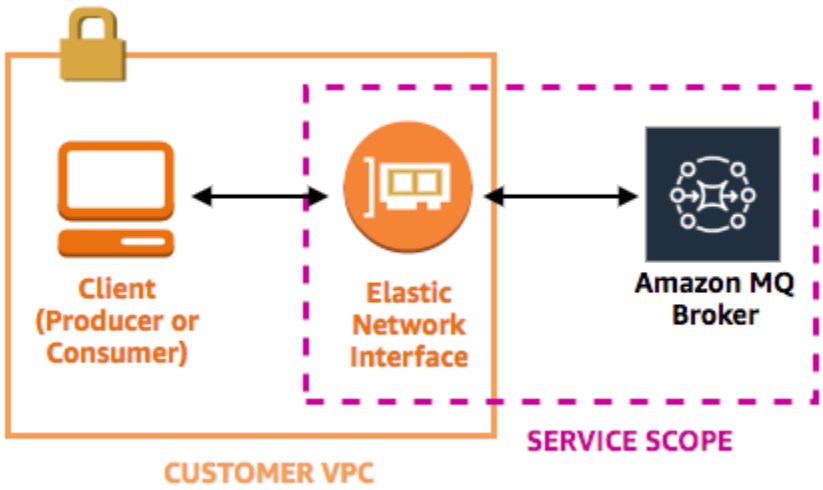
Amazon MQ for ActiveMQ 最佳實務

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。

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 入門：建立並連線至 ActiveMQ 代理程式 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-
west-2.amazonaws.com:61617)?randomize=true
```

> **Important**
>
> 多可用區域代理程式可能會在維護時段和代理程式重新啟動期間遇到容錯移轉。使用容錯移轉傳輸以確保您的代理程式可用性。

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。

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。

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 、Amazon MQ REST API 或 AWS 管理主控台建立、管理和刪除代理程式 AWS Command Line Interface。

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。

什麼是 Amazon MQ for 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。中介裝置執行個體類別 (m7g) 和大小 (large、medium) 的合併描述稱為中介裝置執行個體類型 （ 例如 mq.m7g.large)。

- 單一執行個體代理程式由 Network Load Balancer (NLB) 後方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，請參閱部署 RabbitMQ 代理程式。

接聽程式連接埠

Amazon MQ 受管 RabbitMQ 代理程式支援下列接聽程式連接埠，以便透過 進行應用程式層級連線amqps。您也可以將這些連接埠用於使用 RabbitMQ Web 主控台和管理 API 的用戶端連線。所有連線都使用 TLS 加密來確保安全。

- 接聽程式連接埠 5671 - 用於透過安全 AMQP URL 進行的安全 AMQP 連線。此連接埠支援 RabbitMQ 4 中的 AMQP 0-9-1 和 AMQP 1.0 通訊協定。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 amqps URL：b-c8352341-ec91-4a78-ad9c-a43f23d325bb.mq.us-west-2.amazonaws.com:5671。

- 接聽程式連接埠 443 和 15671 - 您可以交替使用兩個接聽程式連接埠，透過 RabbitMQ Web 主控台或管理 API 存取代理程式。連接埠 443 提供標準 HTTPS 存取，而連接埠 15671 是具有 TLS 加密的傳統 RabbitMQ 管理連接埠。

屬性

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

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` 表示主要版本， `Y` 表示次要版本， `Z`表示修補程式版本編號。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 3.13 版升級至 4.0 版視為主要版本升級。如果只有次要或修補程式版本編號變更，則版本變更會被視為次要變更。例如，從 3.11.28 版升級至 3.12.13 版會被視為次要版本升級。

Amazon MQ for RabbitMQ 建議所有代理程式使用最新的支援版本 RabbitMQ 4.2。如需如何升級代理程式引擎版本的指示，請參閱升級 Amazon MQ 代理程式引擎版本。

當您建立新的 Amazon MQ for RabbitMQ 代理程式時，只需要指定主要和次要版本編號。例如，RabbitMQ 4.2。如果您在建立代理程式時未指定引擎版本，Amazon MQ 會自動預設為最新的引擎版本。

> **Important**
>
> Amazon MQ 不支援串流。建立串流會導致資料遺失。
> Amazon MQ 不支援在 JSON 中使用結構化記錄。

Amazon MQ 支援兩個主要版本的 RabbitMQ：

- RabbitMQ 4

  Amazon MQ RabbitMQ 僅在所有支援的執行個體大小的 mq.m7g 執行個體類型上，支援 RabbitMQ 4 發行系列中的 RabbitMQ 4.2。

- RabbitMQ 3

  Amazon MQ 在所有支援的執行個體大小中，在 mq.t3、mq.RabbitMQ5 和 mq.m7g 執行個體類型上的 RabbitMQ 3.13 發行系列中支援 RabbitMQ 3.13。

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 RabbitMQ 和 `mq.m7g.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.m7g.large`。

RabbitMQ 4

Amazon MQ RabbitMQ 僅在所有支援的執行個體大小的 mq.m7g 執行個體類型上，支援 RabbitMQ 4 發行系列中的 RabbitMQ 4.2。

Amazon MQ 支援從 RabbitMQ 3.13 就地升級至 RabbitMQ 4.2。如需詳細資訊，請參閱從 RabbitMQ 3 升級到 4。

> **Important**
>
> Amazon MQ for RabbitMQ 4.2 代理程式的預設佇列類型為「規定人數」。如果在建立佇列期間未指定佇列類型引數，則會建立規定人數佇列。
> 我們強烈建議在 RabbitMQ 4 上使用規定人數佇列來滿足耐久性需求，因為傳統佇列在所有情況下都無法保證耐用。

下列變更已在 Amazon MQ 上的 RabbitMQ 4 中推出 Amazon MQ

- AMQP 1.0 作為核心通訊協定：如需詳細資訊，請參閱通訊協定。
- Local Shovels：Shovels 現在除了 AMQP 0-9-1 和 AMQP 1.0 之外，還支援稱為 "local" 的新通訊協定。本機 shovel 在內部以 AMQP 1.0 為基礎，但不是使用單獨的 TCP 連線，而是在叢集節點和內部 APIs 之間使用叢集內連線來發佈和取用訊息。這只能用於在相同叢集內消費和發佈，並且可以在使用比 AMQP 0-9-1 和 AMQP 1.0 更少的資源時提供更高的輸送量。
- 配額佇列支援訊息優先順序：配額佇列訊息優先順序一律處於作用中狀態，不需要政策即可運作。一旦規定人數佇列收到具有優先順序設定的訊息，就會啟用優先順序。內部配額佇列僅支援兩個優先順序 - 高和正常。未設定優先順序的訊息將對應至正常，如同優先順序 0 - 4。優先順序高於 4 的訊息將映射至高。高優先順序訊息將以 2：1 的比例優先於一般優先順序訊息，也就是說，每 2 則高優先順序訊息，佇列將提供 1 則一般優先順序訊息（如果有的話）。因此，規定人數佇列會實作一種非嚴格、「公平共享」的優先順序處理。這可確保在正常優先順序訊息上始終進行進度，但高優先順序是以 2：1 的比例有利。
- Khepri：Khepri 用作 RabbitMQ 4 代理程式的預設中繼資料存放區
- 相互 TLS (mTLS)：Amazon MQ 支援 RabbitMQ 代理程式的相互 TLS (mTLS)，允許用戶端使用憑證進行身分驗證。如需詳細資訊，請參閱 mTLS 組態。
- SSL 憑證驗證外掛程式：SSL 身分驗證外掛程式使用來自 mTLS 連線的用戶端憑證來驗證使用者，允許使用 X.509 用戶端憑證進行身分驗證，而非使用者名稱和密碼憑證。如需詳細資訊，請參閱 SSL 憑證身分驗證。

- HTTP 身分驗證外掛程式：HTTP 身分驗證後端外掛程式允許將身分驗證和授權委派給外部 HTTP 服務。如需詳細資訊，請參閱 HTTP 身分驗證和授權。
- JMS 支援：代理程式現在支援已啟用 JMS 主題交換外掛程式的 JMS 工作負載，允許 JMS 應用程式使用 RabbitMQ JMS 用戶端進行連線。

下列功能已從 Amazon MQ 上的 RabbitMQ 4 取代 Amazon MQ

- 鏡射傳統佇列：繼續支援傳統佇列，而不會對用戶端程式庫和應用程式進行任何中斷變更，但它們現在是非複寫佇列類型。用戶端將能夠連線到任何節點，以從任何未複寫的傳統佇列發佈和取用。為了複寫和資料安全，建議使用配額佇列。
- 移除全域 QoS：建議客戶設定每個消費者 QoS （非全域），而不是全域 QoS，其中整個頻道使用單一共用預先擷取。
- 支援暫時性、非獨佔佇列：暫時性佇列是生命週期連結至其宣告節點執行時間的佇列。在單一執行個體代理程式中，它們會在節點重新啟動時移除。在叢集部署中，當其託管的節點重新啟動時，它們會被移除。我們建議在閒置一段時間後，使用佇列 TTL 自動刪除未使用的閒置佇列。系統會繼續支援排入佇列，並在移除佇列的所有連線後刪除。

在 Amazon MQ 上升級至 RabbitMQ 4.2 時，下列重大變更可能會影響您的應用程式

- 預設佇列類型：RabbitMQ 4 代理程式上的預設佇列類型設定為規定人數。如果在建立佇列期間未指定佇列類型引數，則會建立規定人數佇列。
- 規定人數佇列的預設重新傳遞限制設定為 20：重新傳遞 20 次或更多次的訊息將會以無效字母表示或捨棄 （已移除）。如果每則訊息 20 個傳遞是佇列的常見案例，則必須為此類佇列設定無效字母目標或更高限制，以避免資料遺失。建議的做法是透過政策。
- amqplib：早於 0.10.7 的節點 JS 用戶端 amqplib 版本，或任何使用 frame_max < 8192 的 AMQP 用戶端程式庫將無法連線至 RabbitMQ
- 預設資源限制：Amazon MQ for RabbitMQ 已針對連線、頻道、每個頻道的取用者、佇列、虛擬主機、鮑魚、交換和最大訊息大小引入預設資源使用限制。這些可做為保護代理程式可用性的護欄，並且可以使用組態進行自訂，以符合您的特定需求。

Amazon MQ 上的 RabbitMQ 4 不支援下列功能

- 本機隨機交換：Amazon MQ 不支援本機隨機交換，因為 Amazon MQ 節點位於網路負載平衡器後方。
- 訊息攔截器：Amazon MQ 不支援 RabbitMQ 訊息攔截器。 Amazon MQ

- 每個佇列指標：Amazon MQ 不會透過 AWS CloudWatch 為 RabbitMQ 4 代理程式提供 RabbitMQ 佇列指標。Amazon MQ 仍會透過 AWS CloudWatch 提供代理程式層級指標。您可以使用 RabbitMQ 管理 API 查詢佇列指標。我們建議以一分鐘或更長的間隔頻率查詢特定佇列的指標。

RabbitMQ 版本支援

以下 Amazon MQ 版本支援行事曆指出代理程式引擎版本何時會終止支援。當版本終止支援時，Amazon MQ 會自動將此版本的所有代理程式升級至下一個支援的版本。此升級會在代理程式的排程維護時段期間，於end-of-support日期後 45 天內進行。

Amazon MQ 至少會在版本終止支援前 90 天發出通知。我們建議您在end-of-support之前升級代理程式，以防止任何中斷。此外，您無法在支援結束日期的 30 天內，在排程終止支援版本上建立新的代理程式。

RabbitMQ 版本	Amazon MQ 上的終止支援
4.2 （建議）	
3.13	
3.12	2025 年 3 月 17 日

版本升級

您可以隨時手動將代理程式升級至下一個支援的主要或次要版本。如需手動升級代理程式的詳細資訊，請參閱升級 Amazon MQ 代理程式引擎版本。

Amazon MQ 使用 3.13 版及更高版本，管理所有 RabbitMQ 代理程式的最新支援修補程式版本的升級。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。

> **Important**
>
> RabbitMQ 只允許增量版本更新 (例如：3.9.x 至 3.10.x)。您無法在更新時略過次要版本 （例如：3.8.x 到 3.11.x)。

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啟動期間同步。使用較長的佇列時，佇列同步程序可能需要更長的時間。在佇列同步程序期間，消費者和生產者無法使

用佇列。當 queue-sync 程序完成時，代理程式會再次可用。為了將影響降至最低，建議您在低流量時間進行升級。如需版本升級最佳實務的詳細資訊，請參閱 Amazon MQ for RabbitMQ 最佳實踐。

將 Amazon MQ for RabbitMQ 3 代理程式升級至 4

Amazon MQ 支援從 RabbitMQ 3.13 就地升級至 RabbitMQ 4.2。就地升級不需要變更應用程式程式碼。在升級期間，Amazon MQ 會封鎖與代理程式的所有連線。

Amazon MQ 不提供受管藍綠部署選項。如果您選擇獨立執行藍綠部署，請參閱藍綠部署。

> **Important**
>
> 升級之前，請檢閱 RabbitMQ 4 中推出的功能棄用、重大變更和新功能，以確保升級後操作順利進行。

下表比較兩種升級方法。

升級方法的比較

考量事項	就地升級 （建議）	藍綠部署
停機	是，Amazon MQ 會在升級期間封鎖與代理程式的所有連線。停機時間取決於佇列深度。讓佇列保持短暫，可縮短停機時間。	否，您可以將生產者和消費者遷移到新的代理程式，無需停機。
應用程式程式碼變更	不需要變更。升級後，代理程式端點會保持不變。	是，您必須更新您的應用程式程式碼，將生產者和消費者重新導向至新的代理程式。

從 RabbitMQ 3 就地升級至 4

Amazon MQ 支援從 RabbitMQ 3.13 到 RabbitMQ 4.2 的就地主要版本升級。RabbitMQ 4.2 僅支援所有受支援`mq.m7g`執行個體大小的執行個體類型。

> **Note**
>
> 如果您的 Amazon MQ for RabbitMQ 3 代理程式已啟用 Khepri，則 RabbitMQ 4 沒有就地升級路徑。如需詳細資訊，請參閱 RabbitMQ 版本可升級性。在這種情況下，請考慮藍綠部署。

> **Important**
>
> 升級持續時間取決於佇列計數和佇列深度。具有大量佇列和訊息的中介裝置將會經歷更長的停機時間。若要將停機時間降至最低，請讓佇列保持短暫。

步驟 1：升級代理程式執行個體類型

RabbitMQ 4.2 需要`mq.m7g`執行個體類型。如果您的代理程式已在`mq.m7g`執行個體類型上執行，請移至 the section called “步驟 2：將傳統佇列遷移至規定人數佇列”。

使用 UpdateBroker API 操作將代理程式的執行個體類型修改為 `mq.m7g`。

使用 RebootBroker API 重新啟動代理程式以套用執行個體類型變更，或等待下一個排定的維護時段。

如需詳細資訊，請參閱升級 Amazon MQ 代理程式執行個體類型。

步驟 2：將傳統佇列遷移至規定人數佇列

RabbitMQ 4 不支援傳統鏡像佇列。如果代理程式具有傳統佇列或傳統鏡像佇列，Amazon MQ 將防止就地升級至 RabbitMQ 4。

Amazon MQ 提供佇列遷移工具，將傳統佇列遷移至規定人數佇列。此工具可透過管理 > 佇列遷移下的 RabbitMQ Web 主控台，或透過 HTTP API 存取。

若要使用工具，請參閱 Amazon MQ 佇列遷移工具。

步驟 3：將引擎版本從 RabbitMQ 3.13 升級到 4.2

> **Note**
>
> Amazon MQ 會在升級期間封鎖所有傳送至代理程式的外部流量。

> **Important**
>
> 如果您的 RabbitMQ 3.13 叢集代理程式使用客戶受管金鑰 (CMK) 進行加密，則用來呼叫 `UpdateBroker` 以升級至 4.2 版的 IAM 角色必須具有代理程式加密金鑰的下列 AWS KMS 許可：
>
> - `kms:CreateGrant`
> - `kms:DescribeKey`
>
> 如果呼叫角色沒有這些許可，`UpdateBroker`API 會傳回`403`錯誤，指出 AWS KMS 金鑰需要授予許可。若要解決此錯誤，請將 `kms:CreateGrant`和 `kms:DescribeKey`許可新增至代理程式 AWS KMS 金鑰 ARN 的 IAM 角色政策，然後重試升級。

使用 UpdateBroker API 將 RabbitMQ 3.13 代理程式的待處理引擎版本設定為 4.2。

重新啟動代理程式以套用變更，或等待下一個排定的維護時段。

監控升級進度

您可以使用 DescribeBroker API 或 Amazon MQ 主控台上的代理程式隔離狀態來監控升級進度。

Amazon MQ 會在升級開始時執行升級資格檢查。如果識別傳統佇列或啟用 Khepri，Amazon MQ 會將代理程式置於 `CRITICAL_ACTION_REQUIRED` 狀態，並具有動作所需的程式碼 `RABBITMQ_BROKER_NOT_UPGRADEABLE_TO_V4`。Amazon MQ 不會套用主要版本升級，並將讓代理程式可供發佈和使用。

若要繼續升級，請解決基礎問題。如需詳細資訊，請參閱 RABBITMQ_BROKER_NOT_UPGRADEABLE_TO_V4。

升級後更新資源限制組態

Amazon MQ for RabbitMQ 4 為連線、頻道、每個頻道的消費者、佇列、虛擬主機、鑤魚、交換和最大訊息大小引入預設資源限制。在 RabbitMQ 3 代理程式上，這些資源的設定具有最大資源限制。就地升級至 RabbitMQ 4.2 之後，Amazon MQ 會套用 RabbitMQ 4 預設資源限制，低於 RabbitMQ 3 中使用的資源限制上限。

> **Important**
>
> 如果您的 RabbitMQ 3 代理程式使用計數高於 RabbitMQ 4 預設限制的任何資源，代理程式可能會在升級後拒絕超過新限制的新連線、頻道或佇列宣告。升級之前，請檢閱執行個體類型和部署模式的預設資源限制。升級完成後，請更新代理程式組態，以調整資源限制以符合您的工作負載需求。如需詳細資訊，請參閱資源限制組態。

從 RabbitMQ 3 到 4 的藍綠部署

Amazon MQ 不提供從 RabbitMQ 3.13 升級到 RabbitMQ 4.2 的受管藍綠部署選項。如果您選擇獨立執行藍綠部署，此方法需要變更應用程式程式碼，才能將生產者和消費者重新導向至新的代理程式。

如需詳細說明，請參閱 RabbitMQ 文件中的藍綠部署。

Amazon MQ for RabbitMQ 代理程式的部署選項

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)

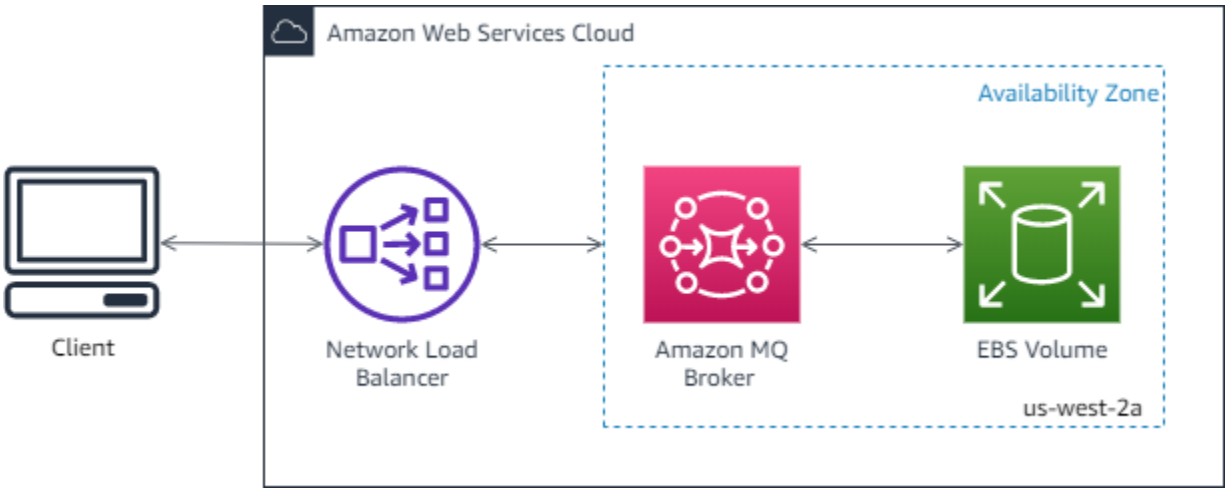
主題

選項 1：Amazon MQ for RabbitMQ 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。



選項 2：Amazon MQ for RabbitMQ 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。

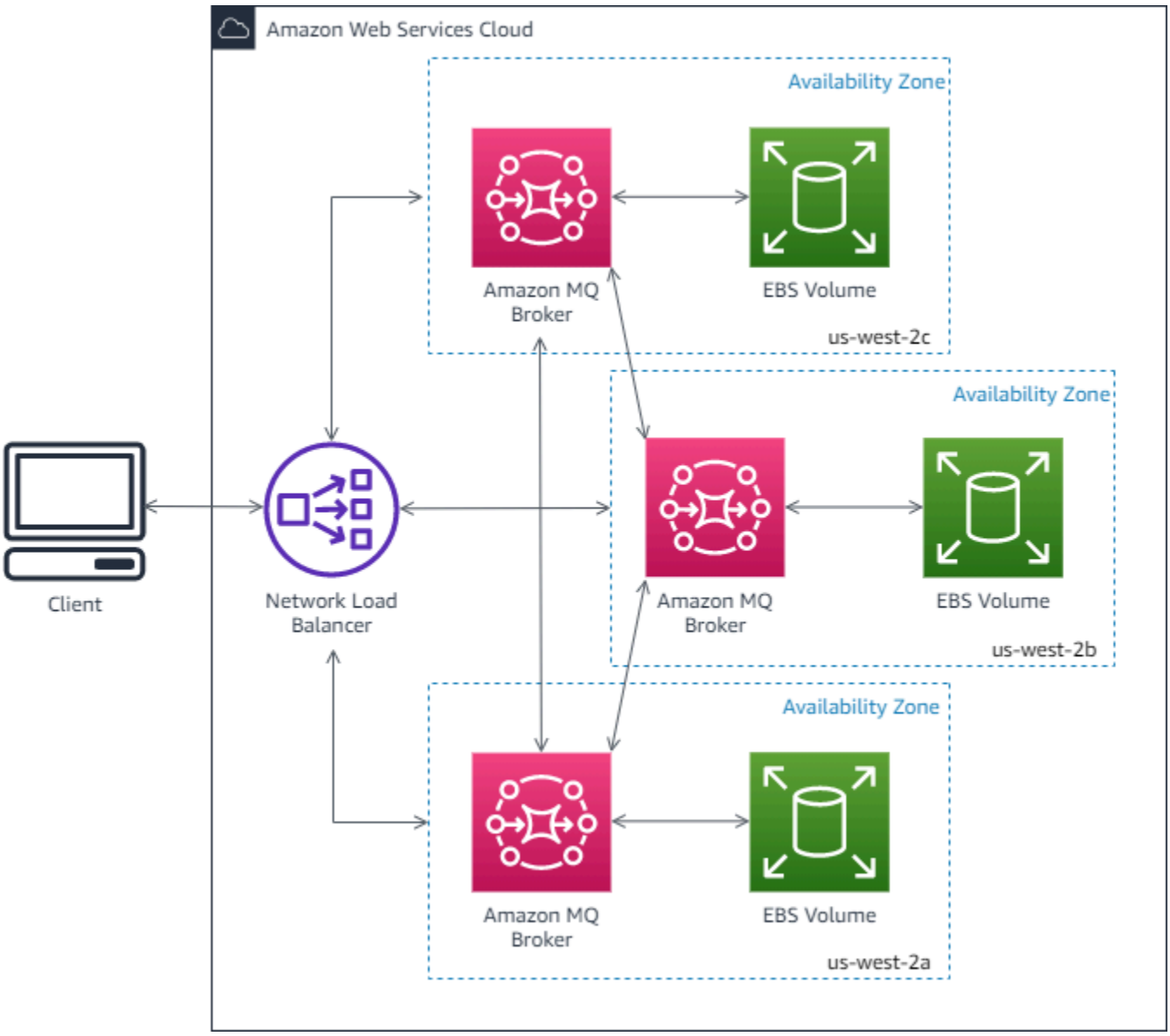
> **Note**
>
> 在叢集部署中，如果發生可用區域中斷，Amazon MQ 會自動嘗試將受影響的 RabbitMQ 節點重新定位到不同的可用區域，以維持叢集大小。一旦中斷解決，叢集會自動重新平衡跨可用 AZs。

> **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 “步驟 1：從網路故障自動復原”。

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 會自動重新建立它。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。

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 代理程式執行個體類型

中介裝置執行個體類別 (m7g) 和大小 （ 大型、中型 ） 的合併描述稱為中介裝置執行個體類型 （ 例如 mq.m7g.large)。

我們建議在叢集和單一執行個體部署中使用 mq.m7g 執行個體類型。

Amazon MQ 至少會在執行個體類型終止支援前 90 天發出通知。我們建議您在end-of-support日期之前將代理程式升級至新的執行個體類型，以防止任何中斷。

> **Important**
>
> 您無法將代理程式從 mq.m7g或 mq.m5 執行個體類型降級為mq.t3.micro執行個體類型。mq.t3.micro 執行個體類型不支援叢集部署。

m7g 叢集部署的執行個體類型

我們建議搭配叢集部署使用mq.m7g.x執行個體類型。下表顯示叢集部署的可用mq.m7g.x執行個體類型。

執行個體類型	vCPU	記憶體 (GiB)	網路基準/高載頻寬 (Gbps)	建議用途	儲存	每個節點的磁碟區大小 (GB)
mq.m7g.medium	1	4	0.52 / 12.5	評估	EBS	5
mq.m7g.large	2	8	0.937 / 12.5	生產	EBS	15
mq.m7g.xlarge	4	16	1.876 / 12.5	生產	EBS	25
mq.m7g.2xlarge	8	32	3.75 / 15.0	生產	EBS	45
mq.m7g.4xlarge	16	64	7.5 / 15.0	生產	EBS	90
mq.m7g.8xlarge	32	128	15 GB	生產	EBS	175
mq.m7g.12xlarge	48	192	22.5 GB	生產	EBS	260
mq.m7g.16xlarge	64	256	30 GB	生產	EBS	345

m7g 單一執行個體部署的執行個體類型

下表顯示單一mq.m7g.x執行個體部署的可用執行個體類型。

執行個體類型	vCPU	記憶體 (GiB)	網路基準/高載頻寬 (Gbps)	建議用途	儲存	每個節點的磁碟區大小 (GB)
mq.m7g.medium	1	4	0.52 / 12.5	評估	EBS	200
mq.m7g.large	2	8	0.937 / 12.5	生產	EBS	200
mq.m7g.xlarge	4	16	1.876 / 12.5	生產	EBS	200
mq.m7g.2xlarge	8	32	3.75 / 15.0	生產	EBS	200
mq.m7g.4xlarge	16	64	7.5 / 15.0	生產	EBS	200
mq.m7g.8xlarge	32	128	15 GB	生產	EBS	200
mq.m7g.12xlarge	48	192	22.5 GB	生產	EBS	200
mq.m7g.16xlarge	64	256	39 GB	生產	EBS	200

mq.m5 單一執行個體部署的執行個體類型

下表顯示單一mq.m5.x執行個體部署的可用執行個體類型

執行個體類型	vCPU	記憶體 (GiB)	網路基準/高載頻寬 (Gbps)	建議用途	儲存	每個節點的磁碟區大小 (GB)
mq.t3.micro	2	1	0.064 / 5.0	評估	EBS	20
mq.m5.large	2	8	0.75 / 10.0	生產	EBS	200
mq.m5.xlarge	4	16	1.25 / 10.0	生產	EBS	200
mq.m5.2xlarge	8	32	2.5 / 10.0	生產	EBS	200
mq.m5.4xlarge	16	64	5.0 / 10.0	生產	EBS	200

mq.m5 叢集部署的執行個體類型

下表顯示叢集部署的可用mq.m5.x執行個體類型

執行個體類型	vCPU	記憶體 (GiB)	網路基準/高載頻寬 (Gbps)	建議用途	儲存	每個節點的磁碟區大小 (GB)
mq.m5.large	2	8	0.75 / 10.0	生產	EBS	200
mq.m5.xlarge	4	16	1.25 / 10.0	生產	EBS	200
mq.m5.2xlarge	8	32	2.5 / 10.0	生產	EBS	200
mq.m5.4xlarge	16	64	5.0 / 10.0	生產	EBS	200

記憶體和磁碟警示

Amazon MQ 會設定每個 RabbitMQ 代理程式的記憶體和磁碟閾值，以防止資源耗盡。超過閾值時，RabbitMQ 會觸發警示，並封鎖發佈者傳送訊息。位於不同連線的消費者會繼續正常運作。不過，如果發佈者和消費者共用相同的連線，則消費者也會遭到封鎖。

> **Important**
>
> Amazon MQ 會管理這些閾值，您無法修改這些閾值。當警示條件清除時，發佈者會自動解除封鎖。如需疑難排解資訊，請參閱 the section called “ RABBITMQ_MEMORY_ALARM ”和 the section called “RABBITMQ_DISK_ALARM”。

記憶體警示

`vm_memory_high_watermark` 參數定義 RabbitMQ 代理程式在封鎖發佈者傳送訊息之前可以使用的最大記憶體數量。當記憶體用量超過此閾值時，RabbitMQ 會觸發記憶體警示。如需詳細資訊，請參閱 RabbitMQ 網站上的記憶體警示。

對於`mq.m7g`執行個體類型，Amazon MQ 會設定下列絕對記憶體高浮水印值：

執行個體類型	記憶體高浮水印 (GiB)
mq.m7g.medium	1.8
mq.m7g.large	4.3
mq.m7g.xlarge	9.3
mq.m7g.2xlarge	19.3
mq.m7g.4xlarge	39.4
mq.m7g.8xlarge	79.7
mq.m7g.12xlarge	119.8
mq.m7g.16xlarge	160.1

對於`mq.m5`執行個體類型，Amazon MQ 會將相對記憶體高浮水印設定為 0.4 （ 可用記憶體的 40%)。

mq.m7g 執行個體上較高的記憶體閾值可讓 RabbitMQ 在觸發警示之前使用更多可用的記憶體。如需使用 mq.m7g 執行個體改善效能的詳細資訊，請參閱 AWS 部落格上的使用 AWS Graviton3-based M7g 執行個體改善 Amazon MQ 上的 RabbitMQ 效能。

磁碟警示

disk_free_limit 參數定義 RabbitMQ 節點所需的最小可用磁碟空間量。當任何節點上的可用磁碟空間低於此限制時，RabbitMQ 會觸發磁碟警示，並封鎖發佈者傳送訊息。如需詳細資訊，請參閱 RabbitMQ 網站上的磁碟警示。

對於mq.m7g執行個體類型，Amazon MQ 會設定下列磁碟可用限制。單一執行個體代理程式具有較高的磁碟可用限制來提供額外的保護，因為它們沒有其他節點可在磁碟空間用盡時提供流量。

部署模式	磁碟可用限制 (GiB)
單一執行個體	10
叢集	2

對於mq.m5執行個體類型，Amazon MQ 會設定下列磁碟可用限制。這些值同時適用於單一執行個體和叢集部署。

執行個體類型	磁碟可用限制 (GiB)
mq.m5.large	12
mq.m5.xlarge	20
mq.m5.2xlarge	36
mq.m5.4xlarge	69

由於mq.m7g執行個體的磁碟可用限制較低，相較於同等mq.m5執行個體，有更多佈建的磁碟區可用於訊息儲存。

Amazon MQ for RabbitMQ 大小調整準則

您可以選擇最能支援您應用程式的中介裝置執行個體類型。選擇執行個體類型時，請考慮會影響代理程式效能的因素：

- 用戶端和佇列的數量
- 傳送的訊息量
- 訊息保留在記憶體中
- 備援訊息

`m7g.medium` 建議僅將較小的中介裝置執行個體類型用於測試應用程式效能。我們建議較大的代理程式執行個體類型 `m7g.large`和更高或生產層級的用戶端和佇列、高輸送量、記憶體中的訊息和備援訊息。

> **Important**
>
> 您無法將代理程式從 `mq.m5`或 `mq.m7g` 執行個體類型降級為 `mq.t3.micro`執行個體類型。

請務必測試您的代理程式，以判斷適合您工作負載傳訊需求的執行個體類型和大小。

一律在 RabbitMQ 4 代理程式上使用預設資源限制，以根據 Amazon MQ 最佳實務判斷應用程式的適當執行個體大小。這些預設資源限制是以m7g執行個體類型和規定人數佇列類型為基礎。

- m7g 單一執行個體部署的預設資源限制
- m7g 叢集部署的預設資源限制

您可以將任何限制的值增加到執行個體類型和部署模式定義的最大值。不過，我們強烈建議您在生產環境中使用 之前，先使用增加的值來測試代理程式效能。

- m7g 單一執行個體部署的最大資源限制
- m7g 叢集部署的最大資源限制
- m5 單一執行個體部署的最大資源限制
- m5 叢集部署的最大資源限制
- 錯誤訊息

> **Note**
>
> RabbitMQ 3.13 代理程式沒有預設資源限制，但建議您使用建議的預設值。

預設資源限制

Amazon MQ for RabbitMQ 支援從 RabbitMQ 4 之後設定代理程式資源限制。當您建立代理程式時，Amazon MQ 會自動將預設值套用至這些資源限制。這些預設值可做為護欄，以保護代理程式可用性，同時適應常見的客戶使用模式。您可以透過變更限制組態值來自訂代理程式行為，以更符合您的特定工作負載需求。

進行變更之前，請注意：

> **Important**
>
> 1. 組態變更可能會影響代理程式效能和可用性
> 2. 在變更任何預設組態選項之前了解影響
> 3. 先測試非生產環境中的組態變更
> 4. 套用變更後監控代理程式的運作狀態

> **Important**
>
> 這些組態的預設值和支援範圍會因 RabbitMQ 版本、執行個體類型和代理程式部署模式而有所不同。

> **Important**
>
> 注意：將代理程式與受支援範圍以外的組態值建立關聯或建立代理程式會導致錯誤回應。

若要了解如何自訂代理程式的這些預設資源限制，請參閱 the section called “設定資源限制”。

適用於 RabbitMQ 4.2 代理程式的預設資源限制為

- m7g 單一執行個體部署的預設資源限制

- m7g 叢集部署的預設資源限制

預設資源限制

> **Important**
>
> Amazon MQ for RabbitMQ 3 代理程式，預設值設定為資源限制上限，Amazon MQ 不提供覆寫資源限制組態的功能。

單一執行個體代理程式的預設值

執行個體類型	每個節點的連線數	每個節點的頻道	每個頻道的消費者數	佇列	vhosts	鮑魚	交換	位元組中的訊息大小
mq.m7g.medium	100	500	10	500	10	30	500	524288
mq.m7g.large	1,500	4，500	10	1,000	50	50	1,000	524288
mq.m7g.xlarge	3,000	9,000	10	2,000	100	100	2,000	524288
mq.m7g.2large	6,000	18,000	10	4,000	150	200	4,000	524288
mq.m7g.4large	12,000	36，000	10	8,000	200	400	8,000	524288
mq.m7g.8large	24，000	72，000	10	16,000	250	800	16,000	524288
mq.m7g.1xlarge	36，000	108，000	10	24，000	300	1，200	24，000	524288
mq.m7g.1xlarge	48，000	144，000	10	32,000	350	1,600	32,000	524288

叢集代理程式的預設值

執行個體類型	每個節點的連線數	每個節點的頻道	每個頻道的消費者數	佇列	vhosts	鮑魚	交換	位元組中的訊息大小
mq.m7g.medium	100	300	10	100	10	10	100	524288
mq.m7g.large	500	1500	10	1,000	50	30	1,000	524288
mq.m7g.xarge	1000	3000	10	2,000	100	60	2,000	524288
mq.m7g.2large	2000	6000	10	4,000	150	120	4,000	524288
mq.m7g.4large	4000	12,000	10	8,000	200	240	8,000	524288
mq.m7g.8large	8000	24，000	10	16,000	250	480	16,000	524288
mq.m7g.1xlarge	12000	36，000	10	24，000	300	720	24000	524288
mq.m7g.1xlarge	16,000	48，000	10	32,000	350	960	32,000	524288

Amazon MQ for RabbitMQ 資源上限

您可以設定資源限制，最多可達下表所示的最大值。若要了解如何更新代理程式的資源限制，請參閱 the section called “設定資源限制”。

針對單一執行個體部署使用規定人數佇列的 m7g 大小調整準則

下表顯示單一執行個體代理程式每個執行個體類型的上限值。

執行個體類型	連線	頻道	每個頻道的消費者數	佇列	Vhosts	鮑魚	交換	位元組中的訊息大小
mq.m7g.medium	300	900	1,000	2,500	10	150	12500	134217728
mq.m7g.large	5,000	15,000	1,000	20,000	1500	250	100,000	134217728
mq.m7g.xlarge	10,000	30,000	1,000	30,000	1,500	500	150，000	134217728
mq.m7g.2large	20,000	60,000	1,000	40,000	1,500	1,000	200,000	134217728
mq.m7g.4large	40,000	120,000	1,000	60,000	1,500	2000	300,000	134217728
mq.m7g.8large	80,000	240，000	1,000	80,000	1,500	4000	400,000	134217728
mq.m7g.1xlarge	120,000	360,000	1,000	100,000	1,500	6,000	500,000	134217728
mq.m7g.1xlarge	160,000	480，000	1,000	120,000	1,500	8,000	600，000	134217728

使用用於叢集部署的規定人數佇列來調整 m7g 的大小準則

下表顯示叢集代理程式每個執行個體類型的上限值。

執行個體類型	每個節點的連線數	每個節點的頻道	每個頻道的消費者數	佇列	Vhosts	鮑魚	交換	位元組中的訊息大小
mq.m7g.medium	300	900	1,000	500	10	50	500	134217728

執行個體類型	每個節點的連線數	每個節點的頻道	每個頻道的消費者數	佇列	Vhosts	鮑魚	交換	位元組中的訊息大小
mq.m7g.large	5,000	15,000	1,000	10,000	1,500	150	50,000	134217728
mq.m7g.xlarge	10,000	30,000	1,000	15,000	1,500	300	75,000	134217728
mq.m7g.2large	20,000	60,000	1,000	20,000	1,500	600	100,000	134217728
mq.m7g.4large	40,000	120,000	1,000	30,000	1,500	1200	150，000	134217728
mq.m7g.8large	80,000	240，000	1,000	40,000	1,500	2,400	200,000	134217728
mq.m7g.1xlarge	120,000	360,000	1,000	50,000	1,500	3,600	250,000	134217728
mq.m7g.1xlarge	160,000	480，000	1,000	60,000	1,500	4，800	300,000	134217728

M5 單一執行個體部署的最大資源限制

下表顯示單一執行個體代理程式每個執行個體類型的上限值。

執行個體類型	連線	頻道	每個頻道的消費者數	佇列	Vhosts	鮑魚
m5.large	5,000	15,000	1,000	30,000	1500	250
m5.xlarge	10,000	30,000	1,000	60,000	1500	500
m5.2xlarge	20,000	60,000	1,000	120,000	1500	1,000
m5.4xlarge	40,000	120,000	1000	240，000	1,000	2,000

m5 叢集部署的最大資源限制

下表顯示叢集代理程式每個執行個體類型的上限值。

執行個體類型	佇列	每個頻道的消費者數	鮑魚
m5.large	10,000	1,000	150
m5.xlarge	15,000	1,000	300
m5.2xlarge	20,000	1,000	600
m5.4xlarge	30,000	1,000	1200

每個節點會套用下列連線和頻道限制：

執行個體類型	連線	頻道
m5.large	5000	15,000
m5.xlarge	10,000	30,000
m5.2xlarge	20,000	60,000
m5.4xlarge	40,000	120,000

叢集代理程式的確切限制值可能低於指示值，具體取決於可用節點的數量，以及 RabbitMQ 如何在可用節點之間分配資源。如果您超過限制值，您可以建立新的節點連線，然後再試一次，也可以升級執行個體大小以增加限制上限

錯誤訊息

超過限制時，會傳回下列錯誤訊息。所有值都以**`m7.large`**單一執行個體限制為基礎。

> **Note**
>
> 下列訊息的錯誤代碼可能會根據您使用的用戶端程式庫而變更。

Connection (連線)

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

Channel

```
ConnectionClosedByBroker 1500 "NOT_ALLOWED - number of channels opened on
node 'rabbit@ip-10-0-23-173.us-west-2.compute.internal' has reached the
maximum allowed limit of (15,000)"
```

消費者

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

訊息大小上限

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

Exchange

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

> **Note**
>
> 下列錯誤訊息使用 HTTP Management API 格式。

佇列

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

雪鏟

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

Vhost

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

Amazon MQ for RabbitMQ 代理程式預設值

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 會為 Amazon MQ for RabbitMQ 3 和 RabbitMQ 4 建立不同的代理程式政策和 vhost 限制。以下小節將詳細討論這些差異。

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：

Amazon MQ for RabbitMQ 3：

- 單一執行個體 – `AWS-DEFAULT-POLICY-SINGLE-INSTANCE`
- 叢集部署 – `AWS-DEFAULT-POLICY-CLUSTER-MULTI-AZ` && `AWS-DEFAULT-QUORUM-QUEUES-POLICY-CLUSTER-MULTI-AZ`

Amazon MQ for RabbitMQ 4：

- 單一執行個體 – `AWS-DEFAULT-POLICY-SINGLE-INSTANCE`
- 叢集部署 – `AWS-DEFAULT-POLICY-CLUSTER` && `AWS-DEFAULT-QUORUM-QUEUES-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)。這些設定僅適用於 Amazon MQ for RabbitMQ 3，且不適用於 RabbitMQ 4。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。

RabbitMQ 3 和 RabbitMQ 4 代理程式的設定

- **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

RabbitMQ 3 的特定設定

- **`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

Important

`max-queues` 和 `max-connections` 僅適用於 Amazon MQ for RabbitMQ 3。

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
mq.m7g.medium	單一執行個體	N/A	2,500	100
	叢集	500,000	100	100
mq.m7g.large	單一執行個體	N/A	20,000	5,000
	叢集	8，000，000	10,000	5,000
mq.m7g.xlarge	單一執行個體	N/A	30,000	10,000
	叢集	9，000，000	15,000	10,000
mq.m7g.2xlarge	單一執行個體	N/A	40,000	20,000
	叢集	10,000,000	40,000	20,000
mq.m7g.4xlarge	單一執行個體	N/A	60,000	40,000
	叢集	12，000，000	30,000	40,000
mq.m7g.8xlarge	單一執行個體	N/A	80,000	80,000
	叢集	20,000,000	40,000	80,000
mq.m7g.12xlarge	單一執行個體	N/A	100,000	120,000
	叢集	30，000，000	20,000	120,000
mq.m7g.16xlarge	單一執行個體	N/A	120,000	160,000
	叢集	40，000，000	50,000	160,000

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
m5.large	叢集	8，000，000	10,000	15,000
m5.xlarge	單一執行個體	N/A	30,000	8,000
m5.xlarge	叢集	9，000，000	10,000	20,000
m5.2xlarge	單一執行個體	N/A	60,000	15,000
m5.2xlarge	叢集	10,000,000	10,000	40,000
m5.4xlarge	單一執行個體	N/A	150，000	30,000
m5.4xlarge	叢集	12，000，000	10,000	100,000

設定 RabbitMQ 代理程式

組態包含 Cuttlefish 格式 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。

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 Resource Name (ARN) (arn：aws：mq：us-east-2：123456789012：configuration：c-1234a5b678cd-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

主題

- 建立和套用 RabbitMQ 代理程式組態
- 編輯 Amazon MQ for RabbitMQ 組態修訂
- Amazon MQ 上 RabbitMQ 的可設定值 Amazon MQ
- RabbitMQ 組態中的 ARN 支援

建立和套用 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。

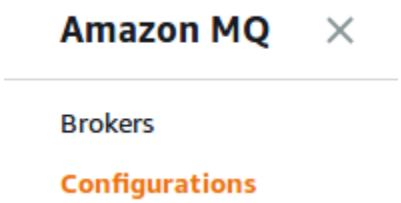
以下範例顯示如何使用 AWS 管理主控台建立和套用 RabbitMQ 代理程式組態。

> **Important**
>
> 您只能使用 `DeleteConfiguration` API 刪除組態。如需詳細資訊，請參閱《Amazon MQ API 參考》中的組態。

建立新組態

若要將組態套用至代理程式，您必須先建立組態。

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

建立組態之後，您必須使用組態修訂來編輯組態。

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。

   ****####****頁面上會顯示新的組態修訂所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)。

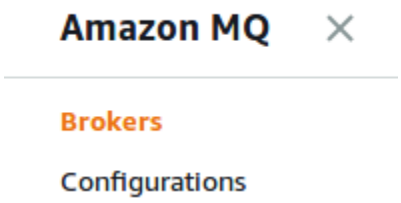
   隨即儲存組態的新修訂版。

> **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。
> 您目前無法刪除組態。

將組態修訂套用至您的代理程式

建立組態修訂之後，您可以將組態修訂套用至代理程式。

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**
>
> 重新啟動時，單一執行個體代理程式處於離線狀態。對於叢集代理程式，代理程式重新啟動時一次只會關閉一個節點。

5. 選擇套用。

   組態修訂會在指定時間套用至代理程式。

編輯 Amazon MQ for RabbitMQ 組態修訂

下列指示說明如何編輯代理程式的組態修訂。

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。

   MyBroker 頁面上會顯示組態所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。
   > 您目前無法刪除組態。

可設定的值

您可以在 中修改代理程式組態檔案，以設定下列代理程式組態選項的值 AWS 管理主控台。

除了下表所述的值之外，Amazon MQ 還支援與身分驗證和授權以及資源限制相關的其他代理程式組態選項。如需這些組態選項的詳細資訊，請參閱

- OAuth 2.0 組態
- LDAP 組態
- HTTP 組態
- SSL 組態
- mTLS 組態
- ARN 支援
- 資源限制
- AMQP 用戶端 SSL 組態

Configuration	預設值	建議值	值	適用的版本	Description
consumer_timeout	1800000 毫秒 (30 分鐘)	1800000 毫秒 (30 分鐘)	0 到 2，147，483，647 毫秒。Amazon MQ 也支援值 0，這表示「無限」。	所有版本	消費者交付確認的逾時，用於偵測消費者何時未處理交付。
活動訊號	60 秒	60 秒	60 到 3600 秒	所有版本	定義 RabbitMQ 認為無法使用連線之前的時間。
management.restrictions.operator_pol	true	true	true、false	所有版本	關閉對運算子政策進行變更。如果您進行此變更，我

Configuration	預設值	建議值	值	適用的版本	Description
icy_changes.disabled					們強烈建議您將 HA 屬性納入您自己的操作員政策中。
quorum_queue.property_equivalence.relaxed_checks_on_redeclaration	true	true	true、false	所有版本	設定為 TRUE 時，您的應用程式會在重新宣告規定人數佇列時避免頻道例外狀況。
secure.management.http.headers.enabled	true	true	true、false	所有版本	開啟無法修改的 HTTP 安全標頭。

設定消費者交付確認

您可以設定 consumer_timeout 來偵測消費者何時不確認交付。如果消費者未在逾時值內傳送確認，頻道將會關閉。例如，如果您使用預設值 1800000 毫秒，如果消費者未在 1800000 毫秒內傳送交付確認，則頻道將會關閉。Amazon MQ 也支援值 0，這表示「無限」。

設定活動訊號

您可以設定活動訊號逾時，找出連線中斷或失敗的時間。活動訊號值會定義連線視為中斷前的時間限制。

設定運算子政策

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 管理主控台 或 Management API 變更運算子政策。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。

在佇列宣告上設定寬鬆檢查

如果您已將傳統佇列遷移至規定人數佇列，但未更新用戶端程式碼，您可以將 quorum_queue.property_equivalence.relaxed_checks_on_redeclaration 設定為 true，以避免在重新宣告規定人數佇列時發生頻道例外狀況。

設定 HTTP 安全標頭

secure.management.http.headers.enabled 組態會啟用下列 HTTP 安全標頭：

- X-Content-Type-Options：nosniff：防止瀏覽器執行內容探查，這是用來推斷網站檔案格式的演算法。
- X-Frame-Options：DENY：防止其他人將管理外掛程式內嵌至自己網站上的影格，以欺騙其他人
- Strict-Transport-Security：max-age=47304000； includeSubDomains：在與網站及其子網域進行長時間 (1.5 年） 的後續連線時， 會強制瀏覽器使用 HTTPS。

在 3.10 版及更高版本上建立的 Amazon MQ for RabbitMQ 代理程式預設會將 secure.management.http.headers.enabled 設為 true。您可以將 secure.management.http.headers.enabled 設定為 true，以開啟這些 HTTP 安全標頭。如果您想要選擇退出這些 HTTP 安全標頭，請將 secure.management.http.headers.enabled 設定為 false。

設定 OAuth 2.0 身分驗證和授權

如需 OAuth 2.0 組態選項和為代理程式設定 OAuth 2.0 身分驗證的相關資訊，請參閱支援的 OAuth 2.0 組態和使用 OAuth 2.0 身分驗證和授權。

設定 LDAP 身分驗證和授權

如需有關 LDAP 組態選項和為代理程式設定 LDAP 身分驗證的資訊，請參閱支援的 LDAP 組態和 使用 LDAP 身分驗證和授權。

設定 HTTP 身分驗證和授權

如需有關 HTTP 身分驗證組態值和為代理程式設定 HTTP 身分驗證的資訊，請參閱 HTTP 身分驗證和授權和 使用 HTTP 身分驗證和授權。

> **Note**
>
> HTTP 身分驗證外掛程式僅適用於 Amazon MQ for RabbitMQ 第 4 版及更新版本。

設定 SSL 憑證身分驗證

如需 SSL 憑證身分驗證組態值和設定代理程式的 SSL 憑證身分驗證的相關資訊，請參閱 SSL 憑證身分驗證和使用 SSL 憑證身分驗證。

> **Note**
>
> SSL 憑證身分驗證外掛程式僅適用於 Amazon MQ for RabbitMQ 第 4 版及更高版本。

設定 mTLS

Amazon MQ for RabbitMQ 支援相互 TLS (mTLS) 以安全連線至各種端點和外部服務。mTLS 要求用戶端和伺服器使用憑證進行身分驗證，以提供增強的安全性。

> **Note**
>
> 針對 mTLS 使用私有憑證授權單位僅適用於 Amazon MQ for RabbitMQ 第 4 版及更高版本。

> **Important**
>
> Amazon MQ for RabbitMQ 強制對憑證和私有金鑰檔案使用 AWS ARNs。如需詳細資訊，請參閱 RabbitMQ 組態中的 ARN 支援。

在此頁面

AMQP 端點

為用戶端連線至 AMQP 端點設定 mTLS。這與 SSL 憑證身分驗證搭配使用。如需支援的組態，請參閱 SSL 憑證身分驗證。

RabbitMQ 管理外掛程式

設定 mTLS 以連線至 RabbitMQ 管理介面。

> **Note**
>
> 管理 API 不支援嚴格 mTLS。

支援的組態

`aws.arns.management.ssl.cacertfile`

憑證授權單位檔案，用於驗證連線至 管理介面的用戶端憑證。

`management.ssl.verify`

對等驗證模式。支援的值：`verify_none`、 `verify_peer`

`management.ssl.depth`

用於驗證的憑證鏈深度上限。

`management.ssl.hostname_verification`

主機名稱驗證模式。支援的值：`wildcard`、 `none`

不支援的 SSL 選項

不支援下列 SSL 組態值：

檢視完整清單

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

RabbitMQ OAuth 2.0 外掛程式

針對從 Amazon MQ 到 OAuth 2.0 身分提供者的連線設定 mTLS。如需支援的組態，請參閱 OAuth 2.0 身分驗證和授權。

RabbitMQ HTTP 身分驗證外掛程式

針對從 Amazon MQ 到 HTTP 身分驗證伺服器的連線設定 mTLS。如需支援的組態，請參閱 HTTP 身分驗證和授權。

RabbitMQ LDAP 外掛程式

針對從 Amazon MQ 到 LDAP 伺服器的連線設定 mTLS。如需支援的組態，請參閱 LDAP 身分驗證和授權。

AMQP 用戶端連線

為聯合和鑰型使用的 AMQP 用戶端連線設定 TLS 對等驗證。如需詳細資訊，請參閱 AMQP 用戶端 SSL 組態。

> **Important**
>
> Amazon MQ 目前不支援設定 AMQP 用戶端連線的用戶端憑證。因此，聯合和鑰波無法連接到需要用戶端憑證身分驗證的已啟用 mTLS 的代理程式。

資源限制組態

Amazon MQ for RabbitMQ 支援從 RabbitMQ 4 之後設定代理程式資源限制。當您建立代理程式時，Amazon MQ 會自動將預設值套用至這些資源限制。這些預設值可做為護欄，保護您的代理程式可用性，同時適應常見的客戶使用模式。您可以透過變更限制組態值來自訂代理程式行為，以更符合您的特定工作負載需求。如需預設和最大允許值的詳細資訊，請參閱 the section called “調整大小準則”。

資源名稱和組態金鑰

資源名稱	組態金鑰
連線	`connection_max`
Channel	`channel_max_per_node`
佇列	`cluster_queue_limit`
Vhost	`vhost_max`
雪鏟	`runtime_parameters.limits.shovel`
Exchange	`cluster_exchange_limit`
每個頻道的消費者數	`consumer_max_per_channel`

資源名稱	組態金鑰
訊息大小上限	`max_message_size`

如何覆寫資源限制

您可以使用 Amazon MQ API 和 Amazon MQ 主控台覆寫資源限制。

下列範例顯示如何使用 覆寫佇列計數預設限制 AWS CLI：

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

成功的調用會建立組態修訂。您必須將組態與 RabbitMQ 代理程式建立關聯，然後重新啟動代理程式以套用覆寫。如需詳細資訊，請參閱 RabbitMQ Broker Configurations

組態中的執行個體特定區段支援

透過 RabbitMQ 4，Amazon MQ 支援組態資料中的區段。區段可讓您在單一組態中定義執行個體特定的資源限制。每個區段對應至特定的執行個體類型和部署模式組合。當您將組態與代理程式建立關聯時，Amazon MQ 會自動套用代理程式執行個體類型和部署模式的相符區段。

> **Important**
>
> 區段支援僅適用於 RabbitMQ 4。如果您嘗試將包含區段的組態套用至 RabbitMQ 3 代理程式，API 會傳回 `BadRequestException`。

區段語法

區段以雙大括號分隔，格式如下：

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

`mode` 值表示部署模式：

- 1 – 單一執行個體代理程式
- 3 – 叢集代理程式

任何其他模式值都是無效的，API 會傳回錯誤。

下列範例顯示具有兩個不同執行個體類型的區段的組態資料：

```
connection_max = 1000

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

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

區段中允許的組態金鑰

區段中僅支援下列資源限制組態金鑰。在區段內新增任何其他組態金鑰會導致 API 錯誤。

- `max_message_size`
- `channel_max_per_node`
- `connection_max`
- `cluster_queue_limit`
- `vhost_max`
- `consumer_max_per_channel`
- `runtime_parameters.limits.shovel`
- `cluster_exchange_limit`

區段優先順序規則

當組態金鑰同時出現在一般（頂層）區段和執行個體特定區段時，稍後出現在組態資料中的值優先。例如，將下列組態套用至`m7g.large`叢集代理程式集`connection_max`至 `2000`：

```
connection_max = 1000

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

反轉順序connection_max設定為 1000，因為一般值是最後的：

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

connection_max = 1000
```

> **Note**
>
> 如果組態資料未定義特定執行個體類型的值，Amazon MQ 會套用預設值。

範例

下列範例示範如何使用 建立具有區段的組態，並將其與代理程式建立關聯 AWS CLI。

使用區段更新組態

執行下列命令，以針對多個執行個體類型更新具有執行個體特定資源限制的組態：

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

此組態定義下列值：

- 一般預設值： connection_max = 1000和 channel_max_per_node = 64

- m7g.large 叢集代理程式： connection_max = 2000和 channel_max_per_node = 128
- m7g.xlarge 叢集代理程式： connection_max = 4000和 channel_max_per_node = 256

將組態與代理程式建立關聯

更新組態後，請將其與您的代理程式建立關聯，然後重新啟動代理程式以套用變更。執行以下命令：

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

資源限制覆寫錯誤

將代理程式與受支援範圍以外的組態值建立關聯或建立代理程式會導致類似以下的錯誤回應：

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

如需執行個體類型和部署模式的預設值和支援範圍上限，請參閱 the section called “預設資源限制”和 the section called “資源限制上限”。

RabbitMQ 組態中的 ARN 支援

Amazon MQ for RabbitMQ 支援某些 RabbitMQ 組態設定值的 AWS ARNs。這是由 RabbitMQ 社群外掛程式 Detectormq-aws 啟用。此外掛程式由 Amazon MQ 開發和維護，也可用於非由 Amazon MQ 管理的自我託管 RabbitMQ 代理程式。 Amazon MQ

⚠ 重要考量

- aws 外掛程式擷取的已解析 ARN 值會在執行時間直接傳遞至 RabbitMQ 程序。它們不會存放在 RabbitMQ 節點上的其他位置。
- Amazon MQ for RabbitMQ 需要 IAM 角色，Amazon MQ 可以擔任該角色來存取設定的 ARNs。這是透過設定 來設定aws.arns.assume_role_arn。
- 呼叫 CreateBroker 或 UpdateBroker APIs的使用者，如果代理程式組態包含 IAM 角色，則必須具有該角色的 iam:PassRole 許可。

- IAM 角色必須存在於與 RabbitMQ 代理程式相同的 AWS 帳戶中。組態中的所有 ARNs 必須與 RabbitMQ 代理程式位於相同的 AWS 區域中。
- Amazon MQ 會在擔任 IAM 角色`aws:SourceArn`時新增 IAM 全域條件式金鑰`aws:SourceAccount`和 。這些值必須用於連接到角色的 IAM 政策中，以進行混淆代理人保護。

在此頁面

支援的金鑰

必要的 IAM 角色

`aws.arns.assume_role_arn`

Amazon MQ 擔任以存取其他 AWS 資源的 IAM 角色 ARN。使用任何其他 ARN 組態時為必要。

AMQP 端點

組態金鑰	說明
`aws.arns.ssl_options.cacertfile`	SSL/TLS 用戶端連線的憑證授權機構檔案。Amazon MQ 需要使用 Amazon S3 或 來存放憑證。

RabbitMQ 管理外掛程式

組態金鑰	說明
`aws.arns.management.ssl.cacertfile`	管理介面 SSL/TLS 連線的憑證授權機構檔案。Amazon MQ 需要使用 Amazon S3 或 來存放憑證。

RabbitMQ OAuth 2.0 外掛程式

組態金鑰	說明
`aws.arns.auth_oauth2.https.cacertfile`	OAuth 2.0 HTTPS 連線的憑證授權單位檔案。Amazon MQ 需要使用 Amazon S3 或 來存放憑證。

RabbitMQ HTTP 身分驗證外掛程式

組態金鑰	說明
`aws.arns.auth_http.ssl_options.cacertfile`	HTTP 身分驗證 SSL/TLS 連線的憑證授權機構檔案。Amazon MQ 需要使用 Amazon S3 或 來存放憑證。
`aws.arns.auth_http.ssl_options.certfile`	Amazon MQ 與 HTTP 身分驗證伺服器之間交互 TLS 連線的憑證檔案。Amazon MQ 需要使用 Amazon S3 或 來存放憑證。
`aws.arns.auth_http.ssl_options.keyfile`	Amazon MQ 與 HTTP 身分驗證伺服器之間交互 TLS 連線的私有金鑰檔案。Amazon MQ 需要使用 AWS Secrets Manager 來存放私有金鑰。

RabbitMQ LDAP 外掛程式

組態金鑰	說明
`aws.arns.auth_ldap.ssl_options.cacertfile`	LDAP SSL/TLS 連線的憑證授權機構檔案。Amazon MQ 需要使用 Amazon S3 或 來存放憑證。
`aws.arns.auth_ldap.ssl_options.certfile`	Amazon MQ 與 LDAP 伺服器之間交互 TLS 連線的憑證檔案。Amazon MQ 需要使用 Amazon S3 或 來存放憑證。

組態金鑰	說明
aws.arns.auth_ldap.ssl_options.keyfile	Amazon MQ 與 LDAP 伺服器之間交互 TLS 連線的私有金鑰檔案。Amazon MQ 需要使用 AWS Secrets Manager 來存放私有金鑰。
aws.arns.auth_ldap.dn_lookup_bind.password	LDAP DN 查詢繫結的密碼。Amazon MQ 需要使用 將密碼 AWS Secrets Manager 儲存為純文字值。
aws.arns.auth_ldap.other_bind.password	LDAP 其他繫結的密碼。Amazon MQ 需要使用 將密碼 AWS Secrets Manager 儲存為純文字值。

IAM 政策範例

如需 IAM 政策範例，包括擔任角色政策文件和角色政策文件，請參閱 CDK 範例實作。

如需如何設定 AWS Secrets Manager 和 Amazon S3 資源的步驟使用 LDAP 身分驗證和授權，請參閱 。

相關中介裝置隔離狀態

如需與 ARN 支援問題相關的中介裝置隔離狀態資訊，請參閱：

- 偵錯工具_INVALID_ASSUMEROLE
- 偵錯工具_INVALID_ARN_LDAP
- RABBITMQ_INVALID_ARN

範例藍本

- 代理程式b-f0fc695e-2f9c-486b-845a-988023a3e55b已設定為使用 IAM 角色<role>來存取 AWS Secrets Manager 秘密 <arn>
- 如果提供給 Amazon MQ 的角色沒有 AWS Secrets Manager 秘密的讀取許可，RabbitMQ 日誌中會顯示下列錯誤：

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

此外，代理程式將進入INVALID_ASSUMEROLE隔離狀態。如需詳細資訊，請參閱 INVALID_ASSUMEROLE。

- LDAP 身分驗證嘗試將會失敗，並顯示下列錯誤：

```
[error] <0.254.0> LDAP bind failed: invalid_credentials
```

- 使用適當的許可修正 IAM 角色

AMQP 用戶端 SSL 組態

聯合和鑰波使用 AMQP 進行上游和下游代理程式之間的通訊。根據預設，Amazon MQ for RabbitMQ 4 中的 AMQP 用戶端會啟用 TLS 對等驗證。透過此設定，在 Amazon MQ 代理程式上執行的聯合和鑰型 AMQP 用戶端將在建立與上游代理程式的連線時執行對等驗證。

在 Amazon MQ 代理程式上執行的 AMQP 用戶端支援與 Mozilla 相同的憑證授權單位。如果您不使用 ACM，請在 Mozilla 包含 CA 憑證清單上使用 CA 發行的憑證。如果您的現場部署代理程式使用來自其他憑證授權單位的憑證，SSL 驗證將會失敗。您可以停用這些使用案例的 TLS 對等驗證。

Important

Amazon MQ 目前不支援設定 AMQP 用戶端連線的用戶端憑證。因此，聯合和鑰波無法連接到需要用戶端憑證身分驗證的已啟用 mTLS 的代理程式。

Important

在 Amazon MQ for RabbitMQ 3 上，AMQP 用戶端的 SSL 屬性是以 RabbitMQ defaults (verify_none) 設定。Amazon MQ for RabbitMQ 3 不支援覆寫這些預設值。

Note

使用預設設定verify_peer，您可以在任何 2 個 Amazon MQ 代理程式之間建立聯合和鮑魚連線，但這不支援在 Amazon MQ 代理程式與私有代理程式或使用非 Amazon MQ CA 憑證執行的內部部署代理程式之間建立連線。若要與私有或內部部署代理程式連線，您需要停用下游 Amazon MQ 代理程式的對等驗證。

AMQP 用戶端 SSL 組態金鑰

Configuration	組態金鑰	支援值
AMQP 用戶端 SSL 對等驗證	`amqp_client.ssl_options.verify`	`verify_none` ，`verify_peer`

如何覆寫 AMQP 用戶端 SSL 對等驗證

您可以使用 RabbitMQ 4 代理程式上的 Amazon MQ API 和 Amazon MQ 主控台覆寫 AMQP 用戶端 SSL 對等驗證。

下列範例顯示如何使用 覆寫 AMQP 用戶端 SSL 對等驗證 AWS CLI：

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

成功的調用會建立組態修訂。您必須將組態與 RabbitMQ 代理程式建立關聯，然後重新啟動代理程式以套用覆寫。如需詳細資訊，請參閱 Creating and applying broker configurations

> **Important**
>
> 使用 時`verify_none`，SSL 加密仍然作用中，但無法驗證對等的身分。僅在必要時使用此設定，並確保您信任目的地代理程式的網路路徑。

Amazon MQ for RabbitMQ 身分驗證和授權

Amazon MQ for RabbitMQ 支援下列身分驗證和授權方法：

簡單身分驗證和授權

在此方法中，代理程式使用者存放在 RabbitMQ 代理程式內部，並透過 Web 主控台或管理 API 進行管理。虛擬主機、交換、佇列和主題的許可會直接在 RabbitMQ 中設定。這是預設方法。如需詳細資訊，請參閱簡易身分驗證和授權。

OAuth 2.0 身分驗證和授權

在此方法中，代理程式使用者及其許可由外部 OAuth 2.0 身分提供者 (IdP) 管理。vhost、交換、佇列和主題的使用者身分驗證和資源許可是透過 OAuth 2.0 供應商的範圍系統集中。這可簡化使用者管理，並啟用與現有身分系統的整合。如需詳細資訊，請參閱 OAuth 2.0 身分驗證和授權。

IAM 身分驗證和授權

在此方法中，代理程式使用者透過 AWS IAM 傳出聯合使用 IAM 憑證進行身分驗證。IAM 登入資料用於從 AWS Security Token Service (STS) 取得 JWT 權杖，而這些 JWT 權杖可作為 OAuth 2.0 權杖進行身分驗證。此方法利用 Amazon MQ for RabbitMQ 中現有的 OAuth 2.0 支援，其中 AWS 做為 OAuth 2.0 身分提供者。使用者身分驗證由 AWS IAM 處理，而 vhost、交換、佇列和主題的資源許可則透過 RabbitMQ 中設定的 IAM 政策和範圍別名進行管理。如需詳細資訊，請參閱 IAM 身分驗證和授權。

LDAP 身分驗證和授權

在此方法中，代理程式使用者及其許可由外部 LDAP 目錄服務管理。使用者身分驗證和資源許可是透過 LDAP 伺服器集中，允許使用者使用其現有的目錄服務憑證來存取 RabbitMQ。如需詳細資訊，請參閱 LDAP 身分驗證和授權。

HTTP 身分驗證和授權

在此方法中，代理程式使用者及其許可由外部 HTTP 伺服器管理。使用者身分驗證和資源許可是透過 HTTP 伺服器集中，允許使用者使用自己的身分驗證和授權提供者來存取 RabbitMQ。如需此方法的詳細資訊，請參閱 HTTP 身分驗證和授權。

SSL 憑證身分驗證

Amazon MQ 支援 RabbitMQ 代理程式的交互 TLS (mTLS)。SSL 身分驗證外掛程式使用來自 mTLS 連線的用戶端憑證來驗證使用者。在此方法中，代理程式使用者會使用 X.509 用戶端憑證進行身分驗證，而不是使用者名稱和密碼憑證。用戶端的憑證會根據信任的憑證授權單位 (CA) 進行驗證，使用者名稱會從憑證中的欄位擷取，例如通用名稱 (CN) 或主體別名 (SAN)。此方法提供強式身分驗證，無需透過網路傳輸登入資料。如需詳細資訊，請參閱 SSL 憑證身分驗證。

> **Note**
>
> RabbitMQ 支援多個同時使用的身分驗證和授權方法。例如，您可以同時啟用 OAuth 2.0 和簡單（內部）身分驗證。如需詳細資訊，請參閱啟用 OAuth 2.0 和簡單（內部）身分驗證的 OAuth 2.0 教學課程一節，以及 RabbitMQ 存取控制文件。 OAuth

簡單身分驗證和授權

Amazon MQ for RabbitMQ 代理程式使用者

> **Note**
>
> 本主題說明使用 RabbitMQ 的預設內部身分驗證和授權機制來管理代理程式使用者。如需所有支援的身分驗證和授權方法的資訊，請參閱 Amazon MQ for RabbitMQ 身分驗證和授權。

每個 AMQP 0-9-1 用戶端連線都有相關聯的使用者。此使用者必須經過身分驗證。每個用戶端連線也會以虛擬主機 (vhost) 為目標。使用者必須擁有此 vhost 的一組許可。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。您可以在建立連線時指定使用者登入資料和目標 vhost。

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**
>
> Amazon MQ for RabbitMQ 不支援使用者名稱「訪客」，並且會在您建立新的代理程式時刪除預設訪客帳戶。Amazon MQ 也會定期刪除任何客戶建立的帳戶，稱為「訪客」。

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。其他 AWS 服務可存取中介裝置使用者名稱，包括 CloudWatch Logs。代理程式使用者名稱不適用於私有或敏感資料。
- 如果您無法存取所有管理員帳戶，請參閱復原代理程式存取權以使用 IAM 身分驗證進行復原。

`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`。

```
PUT /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。

Amazon MQ for RabbitMQ 的 OAuth 2.0 身分驗證和授權

Amazon MQ for RabbitMQ 支援多種身分驗證和授權方法。如需所有支援方法的資訊，請參閱 Amazon MQ for RabbitMQ 代理程式的身分驗證和授權。

在 OAuth 2.0 身分驗證和授權中，代理程式使用者及其許可由外部 OAuth 2.0 身分提供者 (IdP) 管理。vhost、交換、佇列和主題的使用者身分驗證和資源許可是透過 OAuth 2.0 供應商的範圍系統集中。這可簡化使用者管理，並啟用與現有身分系統的整合。

⚠ 重要考量

- Amazon MQ for ActiveMQ 代理程式不支援 OAuth 2.0 整合。
- Amazon MQ for RabbitMQ 不支援私有 CA 發行的伺服器憑證。
- RabbitMQ OAuth 2.0 外掛程式不支援權杖自我檢查端點和不透明存取權杖。它也不會執行字符撤銷檢查。
- 您必須包含 IAM 許可 `mq:UpdateBrokerAccessConfiguration`，才能在現有代理程式上啟用 OAuth 2.0。
- Amazon MQ 會自動建立名為 `monitoring-AWS-OWNED-DO-NOT-DELETE`且具有僅監控許可的系統使用者。即使已啟用 OAuth 2.0 的代理程式，此使用者也會使用 RabbitMQ 的內部身分驗證系統，且僅限於循環介面存取。

如需如何為 Amazon MQ for RabbitMQ 代理程式設定 OAuth 2.0 的詳細資訊，請參閱 使用 OAuth 2.0 身分驗證和授權。

在此頁面

支援的 OAuth 2.0 組態

Amazon MQ for RabbitMQ 支援 RabbitMQ OAuth 2.0 外掛程式中的所有可設定變數，但有下列例外：

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

  由於 Amazon MQ 不支援此金鑰，因此我們不支援 UAA 做為 IdP。
- `management.oauth_resource_servers.{id/index}.oauth_client_secret`
- `auth_oauth2.signing_keys.{id/index}`

OAuth 2.0 身分驗證的其他驗證

Amazon MQ 也會針對 OAuth 2.0 身分驗證強制執行下列額外驗證：

- 所有 URLs都需要以 開頭`https://`。
- 支援的簽章演算法：`Ed25519`、`Ed25519ph`、`Ed448`、`Ed448ph`、`EdDSAES256K`、`ES256`、`ES384`、`ES512`、、`HS256`、`RS384`和 `RS512`。

Amazon MQ for RabbitMQ 的 IAM 身分驗證和授權

Amazon MQ for RabbitMQ 支援多種身分驗證和授權方法。如需所有支援方法的資訊，請參閱 Amazon MQ for RabbitMQ 代理程式的身分驗證和授權。

IAM 身分驗證和授權允許代理程式使用者透過 AWS IAM 傳出聯合使用 IAM 憑證進行身分驗證。在此方法中，IAM 登入資料用於從 AWS Security Token Service (STS) 取得 JWT 字符。這些 JWT 權杖做為身分驗證的 OAuth 2.0 權杖，利用 Amazon MQ for RabbitMQ 中現有的 OAuth 2.OAuth 0 支援，其中 AWS 做為 OAuth 2.0 身分提供者。 AWS IAM 會處理使用者身分驗證，而虛擬主機、交換、佇列和主題的資源許可是透過 RabbitMQ 中設定的 IAM 政策和範圍別名來管理。

⚠ 重要考量

- RabbitMQ 3.13、4.2 版及更新版本支援 IAM 身分驗證。Amazon MQ for ActiveMQ 代理程式不支援此功能。
- IAM 身分驗證需要在您的 AWS 帳戶中設定和使用 IAM 傳出聯合。
- 此方法以 Amazon MQ for RabbitMQ 中現有的 OAuth 2.0 基礎設施為基礎， AWS 並做為 OAuth 2.0 身分提供者。
- Amazon MQ 會自動建立名為 `monitoring-AWS-OWNED-DO-NOT-DELETE`且具有僅監控許可的系統使用者。即使已啟用 IAM 的代理程式，此使用者也會使用 RabbitMQ 的內部身分驗證系統，且僅限於循環介面存取。

在此頁面

IAM 身分驗證的運作方式

Amazon MQ for RabbitMQ 的 IAM 身分驗證使用 IAM 傳出聯合，讓 AWS IAM 憑證能夠向 RabbitMQ 代理程式進行身分驗證。IAM 登入資料用於從 AWS Security Token Service (STS) 取得 JWT 權杖，而這些 JWT 權杖可作為 OAuth 2.0 權杖，以使用 RabbitMQ 代理程式進行身分驗證。

限制

Amazon MQ for RabbitMQ 的 IAM 身分驗證具有下列限制：

- 範圍宣告組態 – 您無法直接使用範圍宣告，因為來自 STS 的 JWT 權杖是巢狀的。金鑰為 `sts.amazonaws.com`，需要在 RabbitMQ 組態中使用範圍別名，將 IAM 角色映射至 RabbitMQ 許可。此限制也會防止完全使用 IAM 政策進行授權，而是需要 RabbitMQ 組態進行授權。

如需有關如何為 Amazon MQ for RabbitMQ 代理程式設定 IAM 身分驗證和授權的資訊，請參閱 使用 IAM 身分驗證和授權。

Amazon MQ for RabbitMQ 的 HTTP 身分驗證和授權

Amazon MQ for RabbitMQ 支援使用外部 HTTP 伺服器的代理程式使用者的身分驗證和授權。如需其他支援的方法，請參閱 Amazon MQ for RabbitMQ 代理程式的身分驗證和授權。

> **Note**
>
> HTTP 身分驗證外掛程式僅適用於 Amazon MQ for RabbitMQ 第 4 版及更新版本。

> **重要考量**
>
> - HTTP 伺服器需要透過公有網際網路存取。Amazon MQ for RabbitMQ 可以設定為使用交互 TLS 驗證 HTTP 伺服器。
> - Amazon MQ for RabbitMQ 會對需要存取本機檔案系統的設定強制使用 AWS ARNs。如需詳細資訊，請參閱 RabbitMQ 組態中的 ARN 支援。
> - 您必須包含 IAM 許可 `mq:UpdateBrokerAccessConfiguration`，才能在現有代理程式上啟用 HTTP 身分驗證。
> - Amazon MQ 會自動建立名為 `monitoring-AWS-OWNED-DO-NOT-DELETE`且具有僅監控許可的系統使用者。即使已啟用 HTTP 的代理程式，此使用者也會使用 RabbitMQ 的內部身

分驗證系統，且僅限於循環介面存取。Amazon MQ 會新增受保護的使用者標籤，以防止刪除此使用者。

如需有關如何為 Amazon MQ for RabbitMQ 代理程式設定 HTTP 身分驗證的資訊，請參閱 使用 HTTP 身分驗證和授權。

在此頁面

- 支援的 HTTP 組態
- Amazon MQ 中 HTTP 組態的其他驗證

支援的 HTTP 組態

Amazon MQ for RabbitMQ 支援 RabbitMQ HTTP 身分驗證外掛程式中的所有可設定變數，但下列例外需要 AWS ARNs 如需 ARN 支援的詳細資訊，請參閱 RabbitMQ 組態中的 ARN 支援。

需要 ARNs組態

`auth_http.ssl_options.cacertfile`

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

`auth_http.ssl_options.certfile`

改用 `aws.arns.auth_http.ssl_options.certfile`

`auth_http.ssl_options.keyfile`

改用 `aws.arns.auth_http.ssl_options.keyfile`

不支援的 SSL 選項

也不支援下列 SSL 組態選項：

檢視完整清單

- `auth_http.ssl_options.cert`
- `auth_http.ssl_options.client_renegotiation`
- `auth_http.ssl_options.dh`

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

Amazon MQ 中 HTTP 組態的其他驗證

Amazon MQ 也會針對 HTTP 身分驗證和授權強制執行下列其他驗證：

- auth_http.http_method 必須是 get或 post
- 下列路徑組態必須使用 HTTPS URLs：
  - auth_http.user_path
  - auth_http.vhost_path
  - auth_http.resource_path
  - auth_http.topic_path
- 如果有任何設定需要使用 AWS ARN，aws.arns.assume_role_arn則必須提供 。

Amazon MQ for RabbitMQ 的 SSL 憑證身分驗證

Amazon MQ for RabbitMQ 支援使用 X.509 用戶端憑證對代理程式使用者進行身分驗證。如需其他支援的方法，請參閱 Amazon MQ for RabbitMQ 代理程式的身分驗證和授權。

> **Note**
>
> SSL 憑證身分驗證外掛程式僅適用於 Amazon MQ for RabbitMQ 第 4 版及更高版本。

> **⚠ 重要考量**
>
> - 用戶端憑證必須由信任的憑證授權機構 (CA) 簽署。Amazon MQ for RabbitMQ 會在身分驗證期間驗證憑證鏈。
> - Amazon MQ for RabbitMQ 會強制將 AWS ARNs用於憑證相關設定，例如 CA 憑證和需要存取本機檔案系統的設定。如需詳細資訊，請參閱 RabbitMQ 組態中的 ARN 支援。
> - Amazon MQ 會自動建立名為 的系統使用者，`monitoring-AWS-OWNED-DO-NOT-DELETE`並具有僅監控許可。即使已啟用 SSL 憑證的代理程式，此使用者也會使用 RabbitMQ 的內部身分驗證系統，且僅限於循環介面存取。Amazon MQ 會新增受保護的使用者標籤，以防止刪除此使用者。

如需有關如何為 Amazon MQ for RabbitMQ 代理程式設定 SSL 憑證身分驗證的資訊，請參閱 使用 SSL 憑證身分驗證。

在此頁面

支援的 SSL 組態

Amazon MQ for RabbitMQ 支援用戶端連線的 SSL/TLS 組態。如需 ARN 支援的詳細資訊，請參閱 RabbitMQ 組態中的 ARN 支援。

需要 ARNs組態

`ssl_options.cacertfile`

改用 `aws.arns.ssl_options.cacertfile`

SSL 憑證登入組態

下列組態控制如何從用戶端憑證擷取使用者名稱：

`ssl_cert_login_from`

指定要用於擷取使用者名稱的憑證欄位。支援的值為：

- `distinguished_name` - 使用完整的辨別名稱
- `common_name` - 使用通用名稱 (CN) 欄位
- `subject_alternative_name` 或 `subject_alt_name` - 使用主體別名

`ssl_cert_login_san_type`

使用主體別名時，請指定 SAN 類型。支援的值：`dns`、`ip`、`email`、`uri`、 `other_name`

`ssl_cert_login_san_index`

使用主體別名時， 會指定要使用的 SAN 項目索引 （以零為基礎）。必須是非負整數。

用戶端連線的 SSL 選項

下列 SSL 選項適用於用戶端連線：

`ssl_options.verify`

對等驗證模式。支援的值：`verify_none`、 `verify_peer`

`ssl_options.fail_if_no_peer_cert`

如果用戶端不提供憑證，是否拒絕連線。布林值。

`ssl_options.depth`

用於驗證的憑證鏈深度上限。

`ssl_options.hostname_verification`

主機名稱驗證模式。支援的值：`wildcard`、 `none`

不支援的 SSL 選項

不支援下列 SSL 組態選項：

檢視完整清單

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

Amazon MQ 中 SSL 組態的其他驗證

Amazon MQ 也會針對 SSL 憑證身分驗證強制執行下列額外驗證：

- 如果有任何設定需要使用 AWS ARN，`aws.arns.assume_role_arn`則必須提供 。

Amazon MQ for RabbitMQ 的 LDAP 身分驗證和授權

Amazon MQ for RabbitMQ 支援使用外部 LDAP 伺服器的代理程式使用者的身分驗證和授權。如需其他支援的方法，請參閱 Amazon MQ for RabbitMQ 代理程式的身分驗證和授權。

⚠ 重要考量

- LDAP 伺服器需要透過公有網際網路存取。Amazon MQ for RabbitMQ 可以設定為使用交互 TLS 驗證 LDAP 伺服器。
- Amazon MQ for RabbitMQ 會針對密碼等敏感 LDAP 設定，以及需要存取本機檔案系統的設定，強制使用 AWS ARNs。如需詳細資訊，請參閱 RabbitMQ 組態中的 ARN 支援。
- 您必須包含 IAM 許可 `mq:UpdateBrokerAccessConfiguration`，才能在現有代理程式上啟用 LDAP。
- Amazon MQ 會自動建立名為 `monitoring-AWS-OWNED-DO-NOT-DELETE`且具有僅監控許可的系統使用者。即使已啟用 LDAP 的代理程式，此使用者也會使用 RabbitMQ 的內部身分驗證系統，且僅限於循環介面存取。Amazon MQ 會新增受保護的使用者標籤，以防止刪除此使用者。

如需有關如何為 Amazon MQ for RabbitMQ 代理程式設定 LDAP 的資訊，請參閱 使用 LDAP 身分驗證和授權。

在此頁面

支援的 LDAP 組態

Amazon MQ for RabbitMQ 支援 RabbitMQ LDAP 外掛程式中的所有可設定變數，但下列例外需要 AWS ARNs 如需 ARN 支援的詳細資訊，請參閱 RabbitMQ 組態中的 ARN 支援。

需要 ARNs組態

`auth_ldap.dn_lookup_bind.password`

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

`auth_ldap.other_bind.password`

改用 `aws.arns.auth_ldap.other_bind.password`

`auth_ldap.ssl_options.cacertfile`

改用 `aws.arns.auth_ldap.ssl_options.cacertfile`

auth_ldap.ssl_options.certfile

改用 aws.arns.auth_ldap.ssl_options.certfile

auth_ldap.ssl_options.keyfile

改用 aws.arns.auth_ldap.ssl_options.keyfile

不支援的 SSL 選項

也不支援下列 SSL 組態選項：

檢視完整清單

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

Amazon MQ 中 LDAP 組態的其他驗證

Amazon MQ 也會針對 LDAP 身分驗證和授權強制執行下列其他驗證：

- `auth_ldap.log` 無法設定為 `network_unsafe`
- LDAP 伺服器必須使用 LDAPS。`auth_ldap.use_ssl` 或 `auth_ldap.use_starttls` 必須明確啟用
- 如果有任何設定需要使用 AWS ARN，`aws.arns.assume_role_arn`則必須提供 。
- `auth_ldap.servers` 必須是有效的 IP 地址或有效的 FQDN
- 下列金鑰必須是有效的 LDAP 辨別名稱：
  - `auth_ldap.dn_lookup_base`
  - `auth_ldap.dn_lookup_bind.user_dn`
  - `auth_ldap.other_bind.user_dn`
  - `auth_ldap.group_lookup_base`

外掛程式

Amazon MQ for RabbitMQ 也支援下列外掛程式。

- RabbitMQ 管理外掛程式
- Shovel 外掛程式
- 聯合外掛程式
- 一致的雜湊交換外掛程式
- OAuth 2 外掛程式
- LDAP 外掛程式
- HTTP 外掛程式
- SSL 憑證外掛程式
- aws 外掛程式
- JMS 主題交換外掛程式
- Prometheus 外掛程式

RabbitMQ 管理外掛程式

Amazon MQ for RabbitMQ 支援 RabbitMQ 管理外掛程式，該外掛程式提供 HTTP 型管理 API，以及 RabbitMQ Web 主控台的瀏覽器型 UI。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。

Shovel 外掛程式

Amazon MQ for RabbitMQ 支援 RabbitMQ shovel 外掛程式，可讓您將訊息從佇列和代理程式上的交換移至另一個代理程式。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。

> **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。Amazon MQ 不支援使用靜態 shovel。

僅支援動態鮑魚。動態 shovel 是使用執行時間參數來設定，並且可以透過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，您可以建立對下列 API 端點的 PUT 請求，以設定動態鏟波。在此範例中，{vhost} 可以取代為代理程式 vhost 的名稱，而 {name} 可以取代為新動態鏟的名稱。

```
/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。以下範例會在 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鏟。同樣地，您可以使用 src-exchange 和 dest-exchange 參數來設定兩個交換之間的鏟。

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

聯合外掛程式

Amazon MQ 支援使用 RabbitMQ 聯合外掛程式的聯合交換和佇列。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。聯合佇列和交換使用點對點連結來連線到其他代理程式中的對等節點。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，您可以使用下列請求內文建立對 /api/parameters/federation-upstream/%2f/myupstream 的 POST 請求。

  ```
  {"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，您可以使用下列請求內文建立對 /api/policies/%2f/federate-me 的 POST 請求。

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

  > **Note**
  >
  > 請求內文假設伺服器上的交換是以 amq 開頭命名。使用規則表達式 ^amq\。 將確保針對名稱開頭為 "amq" 的所有交換啟用聯合。RabbitMQ 伺服器上的交換可以不同的方式命名。

一致雜湊交換外掛程式

Amazon MQ for RabbitMQ 支援 RabbitMQ 一致性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。

對於繫結到一致雜湊交換的佇列，繫結索引鍵為數字即字串，可決定每個佇列的繫結權重。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。

OAuth 2.0 外掛程式

Amazon MQ for RabbitMQ 支援 OAuth 2 身分驗證後端外掛程式。此外掛程式會根據您的代理程式組態有條件地啟用。啟用時，此外掛程式會提供與外部 OAuth 2.0 身分提供者整合的 OAuth 2.0 身分驗證和授權，以進行集中式使用者管理和存取控制。如需 OAuth 2.0 身分驗證的詳細資訊，請參閱 OAuth 2.0 身分驗證和授權。

LDAP 外掛程式

Amazon MQ for RabbitMQ 支援 LDAP 身分驗證後端外掛程式。此外掛程式會根據您的代理程式組態有條件地啟用。啟用時，此外掛程式會提供 LDAP 身分驗證和授權，與外部 LDAP 目錄服務整合，以進行集中式使用者身分驗證和授權。如需 LDAP 身分驗證的詳細資訊，請參閱 LDAP 身分驗證和授權。

HTTP 外掛程式

Amazon MQ for RabbitMQ 支援 HTTP 身分驗證後端外掛程式。此外掛程式會根據您的代理程式組態有條件地啟用。啟用時，此外掛程式會提供 HTTP 身分驗證和授權，與外部 HTTP 伺服器整合，以進行集中式使用者身分驗證和授權。如需 HTTP 身分驗證的詳細資訊，請參閱 HTTP 身分驗證和授權。

> **Note**
>
> HTTP 身分驗證外掛程式僅適用於 Amazon MQ for RabbitMQ 第 4 版及更新版本。

SSL 憑證外掛程式

Amazon MQ 支援 RabbitMQ 代理程式的交互 TLS (mTLS)。SSL 身分驗證外掛程式使用來自 mTLS 連線的用戶端憑證來驗證使用者。此外掛程式會根據您的代理程式組態有條件地啟用。啟用時，它會使用 X.509 用戶端憑證提供憑證型身分驗證，以進行強式身分驗證，而不會透過網路傳輸憑證。如需 SSL 憑證身分驗證的詳細資訊，請參閱 SSL 憑證身分驗證。

> **Note**
>
> SSL 憑證身分驗證外掛程式僅適用於 Amazon MQ for RabbitMQ 第 4 版及更高版本。

aws 外掛程式

aws 外掛程式由 Amazon MQ for RabbitMQ 根據您的代理程式組態有條件地啟用。此社群外掛程式由 Amazon MQ 開發和維護，可在 RabbitMQ 組態設定中使用 AWS ARNs 從 AWS 服務安全地擷取憑證和憑證。如需 ARN 支援的詳細資訊，請參閱 ARN support in RabbitMQ configuration。

JMS 主題交換外掛程式

Amazon MQ for RabbitMQ 一律會啟用 JMS Topic Exchange 外掛程式。它可與 RabbitMQ JMS 用戶端搭配使用，以允許新的和現有的 JMS 應用程式連線到 Amazon MQ for RabbitMQ。

> **Note**
>
> JMS Topic Exchange 外掛程式僅適用於 Amazon MQ for RabbitMQ 第 4 版及更新版本。它預設為啟用，但只有在使用 RabbitMQ JMS 用戶端執行 JMS 工作負載時才會啟用。

Prometheus 外掛程式

Prometheus 外掛程式內建於 RabbitMQ 中，且一律在 Amazon MQ for RabbitMQ 代理程式上啟用。從 RabbitMQ 4.2 開始，Amazon MQ 與使用 Prometheus 進行每個節點指標的 RabbitMQ 開放原始碼策略保持一致，而不是使用管理外掛程式進行長期監控。

外掛程式會透過 Prometheus 文字格式的 `/metrics`、 `/metrics/detailed`和 `/metrics/memory-breakdown`端點公開指標。您可以使用 Prometheus 或相容的監控工具來抓取這些端點，以建置儀表板並設定提醒。

Amazon MQ 也會將這些 Prometheus 指標的精選子集發佈至 CloudWatch。如需 CloudWatch 指標的詳細資訊，請參閱Amazon MQ for RabbitMQ 代理程式可用的 CloudWatch 指標。

如需端點詳細資訊、身分驗證和抓取組態，請參閱 存取 Prometheus 指標。

支援的通訊協定

您可以使用 RabbitMQ 支援的任何程式設計語言，以及針對下列任何通訊協定規格啟用 TLS，來存取 RabbitMQ 代理程式：

- AMQP (0-9-1)
- AMQP 1.0

- JMS 1.1
- JMS 2.0
- JMS 3.1

Amazon MQ for RabbitMQ JMS 支援

您現在可以使用 RabbitMQ JMS 用戶端在 Amazon MQ for RabbitMQ 4 上執行 JMS 1.1、2.0 和 3.1 工作負載。 RabbitMQ

RabbitMQ JMS 用戶端

RabbitMQ JMS 用戶端是開放原始碼 JMS 用戶端程式庫，您需要此程式庫將 JMS 應用程式連線至 Amazon MQ RabbitMQ 代理程式。如需詳細資訊，請造訪官方 GitHub 儲存庫。

支援的 JMS 1.1、2.0 和 3.1 APIs

從 Amazon MQ for RabbitMQ 4 之後，外掛程式一律`jms-topic-exchange`會啟用。因此，您可以將 Amazon MQ for RabbitMQ 4 和 RabbitMQ JMS 用戶端用於 JMS 工作負載。支援 JMS 1.1 中定義的所有 JMS APIs但以下除外：

- 不支援伺服器工作階段 APIs。
- 不支援 XA APIs。
- 不支援 JMS 佇列目的地的 JMS 選擇器。
- 不支援 JMS `NoLocal`訂閱屬性。

支援 JMS 2.0 和 JMS 3.1 中新增的所有 APIs除了：

- `JMSProducer.setDeliveryDelay` 不支援 API。

若要進一步了解如何將 JMS 應用程式連線至 Amazon MQ for RabbitMQ 代理程式，請參閱將 JMS 應用程式連線至 Amazon MQ for RabbitMQ 代理程式的教學課程

身分驗證和授權

支援本節列出的所有身分驗證和授權機制。用於使用 JMS 用戶端連線至代理程式的登入資料與使用 AMQP Java 用戶端連線至 RabbitMQ 代理程式相同。

RabbitMQ 上的 AMQP 佇列互通性

您可以使用 RabbitMQ JMS 用戶端將 JMS 訊息傳送至 AMQP 交換，並使用來自 AMQP 佇列的訊息（此功能不支援 JMS 主題）。這可讓您將特定 JMS 工作負載交互操作或遷移至 AMQP 工作負載。如需詳細資訊，請參閱官方用戶端文件。

將政策套用至 Amazon MQ for RabbitMQ

您可以使用 Amazon MQ 建議的預設值來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

在 Amazon MQ for RabbitMQ 引擎 3.13 版及更新版本上，目前的預設運算子政策為：

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

在 4.0 版及更高版本上，預設運算子政策已變更為：

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

由於 RabbitMQ 4 不支援傳統佇列鏡像和 HA 政策設定，因此需要此變更。
您無法建立同時套用至傳統鏡像佇列和規定人數佇列的政策。如果您希望政策僅適用於規定人數佇列，則必須將 `--apply-to`設定為 `quorum_queues`。如果您使用傳統鏡像佇列和規定人數佇列，則必須使用 建立單獨的政策`--apply-to:classic_queues`以及規定人數佇列政策。

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。

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 Amazon MQ for RabbitMQ 代理程式使用者。

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`***。根據代理程式的執行個體大小和部署模式，例如 **`8000000`** 適用於 `mq.m7g.large` 叢集，使用 Amazon MQ 建議的值取代 *`number-of-messages`*。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 *number-of-messages*。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
```

Amazon MQ 上 RabbitMQ 的配額佇列 Amazon MQ

配額佇列是由領導者（主要複本）和跟隨者（其他複本）組成的複寫佇列類型。如果領導者無法使用，則仲裁佇列會使用 Raft 共識演算法，透過大多數投票來選擇新的領導者節點，而先前的領導者會降級為相同叢集中的跟隨節點。其餘跟隨者會繼續像以前一樣複寫。由於每個節點位於不同的可用區域，如果一個節點暫時無法使用，則訊息傳遞會繼續進行另一個可用區域中新選擇的領導複本。

配額佇列適用於處理毒物訊息，當訊息失敗並重新排入佇列多次時發生。

如果您符合下列條件，則不應使用規定人數佇列：

- 使用暫時性佇列
- 有長佇列待處理項目
- 排定低延遲的優先順序

若要宣告規定人數佇列，請將 標頭`x-queue-type`設定為 `quorum`。

主題

- 從傳統佇列遷移至 Amazon MQ for RabbitMQ 上的規定人數佇列
- Amazon MQ for RabbitMQ 規定人數佇列的政策組態
- Amazon MQ for RabbitMQ 規定人數佇列的最佳實務

從傳統佇列遷移至 Amazon MQ for RabbitMQ 上的規定人數佇列

您可以將傳統鏡像佇列遷移至 3.13 版或更新版本的 Amazon MQ 代理程式上的規定人數佇列，方法是在相同叢集上建立新的虛擬主機，或遷移到適當位置。

選項 1：使用 Amazon MQ for RabbitMQ 佇列遷移工具，從傳統鏡像佇列遷移至規定人數佇列

Amazon MQ 提供佇列遷移工具，將傳統佇列遷移至規定人數佇列。工具可透過 RabbitMQ Web 主控台、管理員 > 佇列遷移，或透過 HTTP API 存取。

若要使用工具，請參閱 Amazon MQ 佇列遷移工具。

選項 2：使用新的虛擬主機從傳統鏡像佇列遷移到規定人數佇列

您可以在 3.13 版或更新版本的 Amazon MQ 代理程式上將傳統鏡像佇列遷移至規定人數佇列，方法是在相同叢集上建立新的虛擬主機。

1. 在您現有的叢集中，建立預設佇列類型為規定人數的新虛擬主機 (vhost)。
2. 使用傳統鏡像佇列，聯合外掛程式從新的 vhost 建立指向舊 vhost 的 URI。
3. 使用 `rabbitmqadmin`，將定義從舊 vhost 匯出到新檔案。您必須變更結構描述檔案，使其與規定人數佇列相容。如需您需要對檔案進行的完整變更清單，請參閱 RabbitMQ 規定人數佇列文件中的移動定義。將必要的變更套用至檔案後，將定義重新匯入至新的 vhost。

4. 在新的 vhost 中建立新的政策。如需規定人數佇列的 Amazon MQ 政策組態建議，請參閱 Amazon MQ for RabbitMQ 規定人數佇列的政策組態 。然後，啟動您先前從舊 vhost 到新 vhost 建立的聯合。
5. 將消費者和生產者指向新的 vhost。
6. 設定 Shovel 外掛程式以移動任何剩餘的訊息。一旦佇列為空，請刪除 Shovel。

從傳統鏡像佇列遷移到規定人數佇列就位

您可以將傳統鏡像佇列遷移到 3.13 版或更新版本的 Amazon MQ 代理程式上的仲裁佇列。

1. 停止消費者和生產者。
2. 建立新的臨時規定人數佇列。
3. 設定 Shovel 外掛程式，將任何訊息從舊的傳統鏡像佇列移至新的暫時規定人數佇列。將所有訊息移至臨時規定人數佇列後，請刪除 Shovel。
4. 刪除來源傳統鏡像佇列。然後，以與來源傳統鏡像佇列相同的名稱和繫結重新建立規定人數佇列。
5. 建立新的 Shovel，將訊息從暫時規定人數佇列移至新的規定人數佇列。

Amazon MQ for RabbitMQ 規定人數佇列的政策組態

您可以將特定政策組態新增至 Amazon MQ 上 RabbitMQ 代理程式的仲裁佇列。

當您為規定人數佇列建立政策時，必須執行下列動作：

- 移除以 開頭的所有政策屬性`ha`，例如 `ha-mode`、、`ha-paramsha-sync-mode`、`ha-promote-on-shutdown`、 `ha-sync-batch-size`和 `ha-promote-on-failure`。
- 移除 `queue-mode`。
- 設定為 時變更溢位 `reject-publish-dlx`

> **Important**
>
> Amazon MQ for RabbitMQ 會套用政策中的所有屬性，或不套用政策中的所有屬性。您無法建立同時套用至傳統鏡像佇列和規定人數佇列的政策。如果您希望政策僅適用於規定人數佇列，則必須將 `--apply-to`設定為 `quorum_queues`。如果您使用傳統鏡像佇列和規定人數佇列，則必須使用 建立單獨的政策`--apply-to`：`classic_queues` 以及規定人數佇列政策。

您不需要修改AWS-DEFAULT政策，因為它們會自動在「套用」參數中採用新的佇列類型。如需 Amazon MQ for RabbitMQ 預設政策的詳細資訊，請參閱 設定運算子政策。

Amazon MQ for RabbitMQ 規定人數佇列的最佳實務

建議使用下列最佳實務來改善使用規定人數佇列時的效能。

透過設定交付限制來處理毒物訊息

當訊息失敗並重新傳遞多次時，會發生毒訊息。您可以使用delivery-limit政策引數來設定訊息傳遞限制，以捨棄多次重新傳遞的訊息。如果訊息重新傳遞的時間超過傳遞限制允許的時間，則 RabbitMQ 會捨棄並刪除訊息。當您設定交付限制時，訊息會在佇列前端附近重新排入佇列。

規定人數佇列的訊息優先順序

配額佇列沒有訊息優先順序。如果您需要訊息優先順序，則必須建立多個規定人數佇列。如需使用多個規定人數佇列排定訊息優先順序的詳細資訊，請參閱 RabbitMQ 文件中的訊息優先順序。

使用預設複寫係數

對於使用規定人數佇列的叢集代理程式，Amazon MQ for RabbitMQ 預設為三 (3) 個節點的複寫係數。如果您變更 `x-quorum-initial-group-size`，Amazon MQ 會再次預設為複寫係數 3。

Amazon MQ for RabbitMQ 最佳實踐

使用 Amazon MQ for RabbitMQ 代理程式時，請遵循這些生產就緒性準則，將代理程式效能最大化，並最佳化訊息輸送量效率。

> **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。

主題

- Amazon MQ for RabbitMQ 中代理程式設定和連線管理的最佳實務
- Amazon MQ for RabbitMQ 中訊息耐用性和可靠性的最佳實務
- Amazon MQ for RabbitMQ 中效能最佳化和效率的最佳實務

- Amazon MQ for RabbitMQ 中網路彈性和監控的最佳實務

Amazon MQ for RabbitMQ 中代理程式設定和連線管理的最佳實務

中介裝置設定和連線管理是防止中介裝置訊息輸送量、資源使用率和處理生產工作負載能力問題的第一步。建立和設定 Amazon MQ for RabbitMQ 代理程式時，請完成下列最佳實務，以選取適當的執行個體類型、有效管理連線，以及設定訊息預先擷取以最大化代理程式的效能。

> **Important**
>
> Amazon MQ for RabbitMQ 不支援使用者名稱「訪客」，並且會在您建立新的代理程式時刪除預設訪客帳戶。Amazon MQ 也會定期刪除任何客戶建立的帳戶，稱為「訪客」。

步驟 1：使用叢集部署

對於生產工作負載，我們建議您使用叢集部署，而不是單一執行個體代理程式，以確保高可用性和訊息彈性。叢集部署會移除單一故障點，並提供更好的容錯能力。

叢集部署由分佈在三個可用區域的三個 RabbitMQ 代理程式節點組成，提供自動容錯移轉，並確保即使整個可用區域無法使用，操作仍可繼續。Amazon MQ 會自動複寫所有節點的訊息，以確保節點故障或維護期間的可用性。

叢集部署對於生產環境至關重要，且受 Amazon MQ 服務水準協議支援。

如需詳細資訊，請參閱 Amazon MQ for RabbitMQ 中的叢集部署。

步驟 2：選擇正確的代理程式執行個體類型

中介裝置執行個體類型的訊息輸送量取決於您的應用程式使用案例。 `M7g.medium`應僅用於測試應用程式效能。在生產環境中使用較大的執行個體之前，使用此較小的執行個體可以改善應用程式效能。在執行個體類型 `m7g.large` 和更高版本上，您可以使用叢集部署以獲得高可用性和訊息耐久性。較大的中介裝置執行個體類型可以處理用戶端和佇列的生產層級、高輸送量、記憶體中的訊息和備援訊息。

如需選擇正確執行個體類型的詳細資訊，請參閱 Amazon MQ for RabbitMQ 中的調整大小準則。

步驟 3：使用規定人數佇列

對於 3.13 及更高版本的 RabbitMQ 代理程式，在生產環境中複寫佇列類型的預設選擇應該是使用叢集部署的配額佇列。配額佇列是一種現代複寫佇列類型，可提供高可靠性、高輸送量和穩定的延遲。

配額佇列使用 Raft 共識演算法來提供更好的容錯能力。當領導節點無法使用時，仲裁佇列會自動透過多數投票來選擇新的領導，確保訊息傳遞以最小的中斷繼續。由於每個節點位於不同的可用區域，即使整個可用區域暫時無法使用，您的簡訊系統仍會保持可用狀態。

若要宣告規定人數佇列，請在建立佇列`x-queue-type`時將 標頭設定為 `quorum`。

如需規定人數佇列的詳細資訊，包括遷移策略和最佳實務，請參閱 Amazon MQ for RabbitMQ 中的規定人數佇列。

步驟 4：使用多個頻道

若要避免連線流失，請透過單一連線使用多個頻道。應用程式應避免 1：1 連線到頻道比率。我們建議針對每個程序使用一個連線，然後針對每個執行緒使用一個頻道。避免過度使用頻道，以防止頻道洩漏。

Amazon MQ for RabbitMQ 中訊息耐用性和可靠性的最佳實務

將應用程式移至生產環境之前，請完成下列最佳實務，以防止訊息遺失和資源過度使用。

步驟 1：使用持久性訊息和持久性佇列

持久性訊息有助於在代理程式當機或重新啟動的情況下保護資料耐久性。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。使用 AMQP 0-9-1 時，您可以透過設定交付模式 "2"，將訊息標記為持久性。

```
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());
```

步驟 2：設定發佈者確認和消費者交付確認

確認訊息已傳送至代理程式的程序稱為發佈者確認。發佈者確認讓應用程式知道訊息何時可靠地存放。發佈者確認也有助於控制存放在代理程式的訊息速率。如果沒有發佈者確認，則無法確認訊息已成功處理，而且您的代理程式可能會捨棄無法處理的訊息。

同樣地，當用戶端應用程式將訊息的交付確認和取用傳回給代理程式時，稱為消費者交付確認。確認和確認對於確保使用 RabbitMQ 代理程式時的資料安全至關重要。

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，可以透過設定 `basic.consume`方法啟用確認。AMQP 0-9-1 用戶端也可以透過傳送 `confirm.select`方法來設定發佈者確認。

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。

您可以設定 consumer_timeout來偵測消費者何時不確認交付。如果消費者未在逾時值內傳送確認，則頻道將會關閉，而您將會收到 PRECONDITION_FAILED。若要診斷錯誤，請使用 UpdateConfiguration API 來增加consumer_timeout值。

步驟 3：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。

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。

您也可以定義自動刪除和 TTL 政策，以主動減少資源用量，並將消費者NACKs 保持在最低限度。在代理程式上重新排入佇列訊息是 CPU 密集型，因此大量的 NACKs 可能會影響代理程式效能。

Amazon MQ for RabbitMQ 中效能最佳化和效率的最佳實務

您可以將 Amazon MQ for RabbitMQ 代理程式效能最佳化，方法是將輸送量最大化、將延遲降至最低，並確保有效率的資源使用率。完成下列最佳實務以最佳化您的應用程式效能。

步驟 1：將訊息大小保持在 1 MB 以下

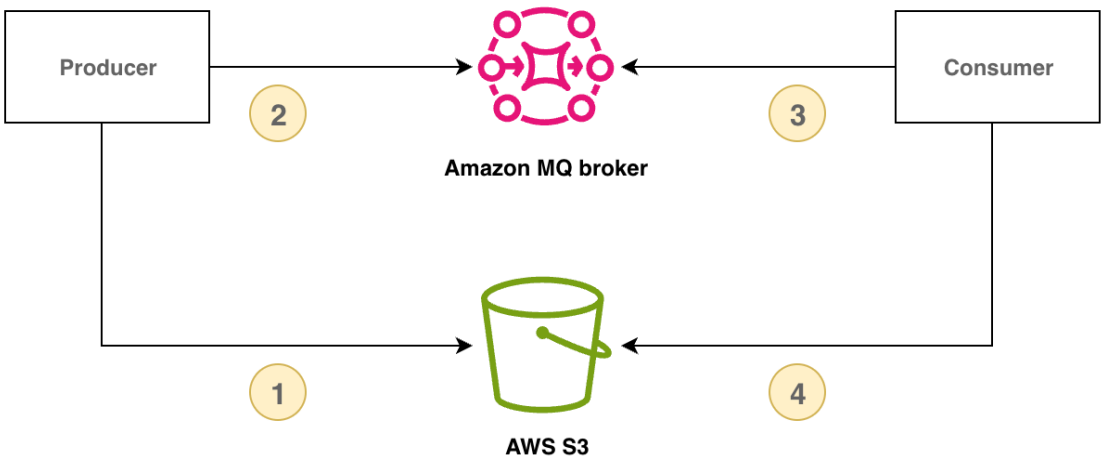
我們建議將訊息保持在 1 MB (MB) 以下，以獲得最佳效能和可靠性。

RabbitMQ 3.13 預設支援高達 128 MB 的訊息大小，但大型訊息可能會觸發無法預測的記憶體警示，封鎖發佈並可能建立高記憶體壓力，同時跨節點複寫訊息。過大的訊息也可能會影響代理程式重新啟動和復原程序，這會提高服務持續性的風險，並可能導致效能降低。

使用宣告檢查模式存放和擷取大型承載

若要管理大型訊息，您可以將訊息承載存放在外部儲存體中，並僅透過 RabbitMQ 傳送承載參考識別符，以實作宣告檢查模式。消費者使用承載參考識別符來擷取和處理大型訊息。

下圖示範如何使用 Amazon MQ for RabbitMQ 和 Amazon S3 實作宣告檢查模式。



下列範例示範此模式使用 Amazon MQ、AWS 適用於 Java 的 SDK 2.x 和 Amazon S3：

1. 首先，定義將保留 Amazon S3 參考識別符的訊息類別。

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

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

2. 建立發佈者方法，將承載存放在 Amazon S3 中，並透過 RabbitMQ 傳送參考訊息。

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

    // Send the reference through RabbitMQ.
```

```
    Message message = new Message();
    message.s3Key = s3Key;
    message.s3Bucket = S3_BUCKET;
    // Assign values to other fields in your message instance.

    publishMessage(message);
}
```

3. 實作取用者方法，從 Amazon S3 擷取承載、處理承載，以及刪除 Amazon S3 物件。

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

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

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

步驟 2：使用 **basic.consume**和長期消費者

basic.consume 搭配長期消費者使用 比使用 輪詢個別訊息更有效率basic.get。如需詳細資訊，請參閱輪詢個別訊息。

步驟 3：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)`。

步驟 4：搭配規定人數佇列使用 Celery 5.5 或更新版本

Python Celery 是分散式任務佇列系統，可在遇到高任務負載時產生許多非關鍵訊息。這項額外的代理程式活動可能會觸發the section called “ RABBITMQ_MEMORY_ALARM ”並導致代理程式無法使用。若要降低觸發記憶體警示的機會，請執行下列動作：

對於所有 Celery 版本

1. 關閉 `task_create_missing_queues` 以減輕佇列流失。

2. 然後，關閉 `worker_enable_remote_control`以停止動態建立`celery@...pidbox`佇列。這將減少代理程式上的佇列流失。

```
worker_enable_remote_control = false
```

3. 若要進一步減少非關鍵訊息活動，請在啟動 Celery 應用程式時，透過不包含 `-E`或 `--task-events`旗標來關閉 Celery worker-send-task-events。
4. 使用下列參數啟動您的 Celery 應用程式：

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

對於 Celery 5.5 版及更新版本

1. 升級至 Celery 5.5 版、支援規定人數佇列的最低版本，或更新版本。若要檢查您正在使用的 Celery 版本，請使用 `celery --version`。如需規定人數佇列的詳細資訊，請參閱 the section called “配額佇列”。
2. 升級至 Celery 5.5 或更新版本後，`task_default_queue_type`將 設定為「規定人數」。
3. 然後，您還必須在中介裝置傳輸選項中開啟發佈確認：

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

Amazon MQ for RabbitMQ 中網路彈性和監控的最佳實務

網路彈性和監控代理程式指標對於維護可靠的簡訊應用程式至關重要。完成下列最佳實務，以實作自動復原機制和資源監控策略。

步驟 1：從網路故障自動復原

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。

步驟 2：監控代理程式指標和警示

建議您定期監控 Amazon MQ for RabbitMQ 代理程式的 CloudWatch 指標和警示，以便在潛在問題影響您的簡訊應用程式之前加以識別和解決。主動監控對於維護彈性傳訊應用程式並確保最佳效能至關重要。

Amazon MQ for RabbitMQ 會將指標發佈至 CloudWatch，以提供中介裝置效能、資源使用率和訊息流程的洞見。要監控的關鍵指標包括記憶體用量和磁碟用量。您可以在代理程式接近資源限制或效能降低時設定 CloudWatch 警示。

監控下列基本指標：

`RabbitMQMemUsed` 和 `RabbitMQMemLimit`

監控記憶體用量，以防止可能封鎖訊息發佈的記憶體警示。

`RabbitMQDiskFree` 和 `RabbitMQDiskFreeLimit`

監控磁碟用量，以避免可能導致代理程式故障的磁碟空間問題。

對於叢集部署， 也會監控節點特定的指標，以識別節點特定的問題。

> **Note**
>
> 如需如何防止高記憶體警示的詳細資訊，請參閱解決和防止高記憶體警示。

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 啟用或停用 CloudWatch 日誌 AWS 管理主控台。

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 發行代理程式時將代理程式升級至新版本，請選擇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。本教學課程會引導您完成安裝 Pika、宣告佇列、設定發佈者將訊息傳送到代理程式的預設交換，以及設定取用者從佇列接收訊息。

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 Amazon MQ for RabbitMQ 代理程式使用者。

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>** – 您在其中建立 Amazon MQ for RabbitMQ 代理程式 AWS 的區域。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：建立消費者並接收訊息

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()
```

與您在上一個步驟中建立的發佈者類似，BasicMessageReceiver繼承BasicPikaClient並實作接收單一訊息和關閉連線的其他方法。

2. 在 `if __name__ == "__main__":` 下方，請用您的資訊取代傳遞給 BasicMessageReceiver 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

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 Amazon MQ for RabbitMQ 代理程式使用者。

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。在政策列中，您的佇列應列出您建立的新政策名稱（例如 batch-size-policy)。
3. 若要以較低的批次大小重新啟動同步程序，請先取消佇列同步。然後重新啟動佇列同步。

> **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 Amazon MQ for RabbitMQ 代理程式可用的 CloudWatch 指標。
- 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 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 https://docs.aws.amazon.com//amazon-mq/latest/developer-guide/rabbitmq-defaults.html。

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。

減少連線和通道的數量

用戶端應用程式可以使用 RabbitMQ Web 主控台手動關閉與 Amazon MQ 代理程式 RabbitMQ 的連線。若要使用 RabbitMQ Web 主控台關閉連線，請執行下列動作：

1. 登入 AWS 管理主控台 並開啟代理程式的 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。當您關閉連線時，Amazon MQ 上的 RabbitMQ Amazon MQ 會將您指定的原因傳回給用戶端。
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。

步驟 2：將 JVM 型應用程式連接至您的代理程式

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。

先決條件

> **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. 針對類型，選取自訂 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 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：（選用）連接至 AWS Lambda 函數

AWS Lambda 可以連線至您的 Amazon MQ 代理程式並使用訊息。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 也必須連線到 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 的介面 VPC 端點。

3. 使用 AWS 管理主控台，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 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=="
      }
    ]
  }
}
```

如需將 Amazon MQ 連線至 Lambda、Lambda 支援 Amazon MQ 事件來源的選項，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。

使用 Amazon MQ for RabbitMQ 的 OAuth 2.0 身分驗證和授權

本教學說明如何使用 Amazon Cognito 做為 OAuth 2.0 供應商，為您的 Amazon MQ for RabbitMQ 代理程式設定 OAuth 2.0 身分驗證。

> **Note**
>
> Amazon Cognito 不適用於中國 （北京） 和中國 （寧夏）。

> **Important**
>
> 本教學課程專屬於 Amazon Cognito，但您可以使用其他身分提供者 (IdPs)。如需詳細資訊，請參閱 OAuth 2.0 身分驗證範例。

在此頁面

設定 OAuth 2.0 身分驗證的先決條件

您可以部署 AWS CDK 堆疊、Amazon Cognito RabbitMQ OAuth 2 的 Amazon Cognito 堆疊外掛程式，藉此設定本教學課程中所需的 Amazon Cognito 資源。如果您要手動設定 Amazon Cognito，請確保您在 Amazon MQ for RabbitMQ 代理程式上設定 OAuth 2.0 之前符合下列先決條件：

設定 Amazon Cognito 的先決條件

- 透過建立使用者集區來設定 Amazon Cognito 端點。若要這樣做，請參閱 Amazon Cognito 中名為如何使用 OAuth 2.0 的部落格：了解不同的 OAuth 2.0 授予。
- 在使用者集區`rabbitmq`中建立名為 的資源伺服器，並定義下列範圍：`read:all`、`configure:all`、 `write:all`和 `tag:administrator`。這些範圍將與 RabbitMQ 許可相關聯。

  如需建立資源伺服器的相關資訊，請參閱《Amazon Cognito 開發人員指南》中的為您的使用者集區 (AWS 管理主控台) 定義資源伺服器。
- 建立下列應用程式用戶端：
  - 類型為 之使用者集區的應用程式用戶端`Machine-to-Machine application`。這是機密用戶端，其用戶端秘密將用於 RabbitMQ AMQP 用戶端。如需應用程式用戶端和建立用戶端的詳細資訊，請參閱應用程式用戶端類型和建立應用程式用戶端。
  - 類型為 之使用者集區的應用程式用戶端`Single-page application`。這是公有用戶端，將用於登入 RabbitMQ 管理主控台的使用者。您必須更新此應用程式用戶端，以包含您將在下列程序中建立的 Amazon MQ for RabbitMQ 代理程式端點，做為允許的回呼 URL。如需詳細資訊，請參閱使用 Amazon Cognito 主控台設定受管登入。

設定 Amazon MQ 的先決條件

- 執行 bash 指令碼的工作 Docker 安裝，以驗證 OAuth 2.0 設定是否成功。
- AWS CLI 版本 >= `2.28.23` 以在代理程式建立期間選擇性新增使用者名稱和密碼。

使用 設定 Amazon Cognito 的 OAuth 2.0 身分驗證 AWS CLI

下列程序說明如何使用 Amazon Cognito 做為 IdP，為您的 Amazon MQ for RabbitMQ 代理程式設定 OAuth 2.0 身分驗證。 Amazon Cognito IdP 此程序使用 AWS CLI 來建立和設定必要的資源。

在下列程序中，請務必將 configurationID 和 Revision、*<c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca>* 和 *<2>* 等預留位置值取代為實際值。

1. 使用 create-configuration AWS CLI 命令建立新的組態，如下列範例所示。

```
aws mq create-configuration \
  --name "rabbitmq-oauth2-config" \
  --engine-type "RABBITMQ" \
  --engine-version "3.13"
```

此命令會傳回類似下列範例的回應。

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

2. 建立名為 的組態檔案**rabbitmq.conf**，以使用 OAuth 2.0 做為身分驗證和授權方法，如下列範例所示。

```
auth_backends.1 = oauth2

# FIXME: Update this value with the token signing key URL of your Amazon Cognito
 user pool.
# If you used the AWS CDK stack to deploy Amazon Cognito, this is one of the stack
 outputs.
auth_oauth2.jwks_url = ${RabbitMqOAuth2TestStack.JwksUri}
```

```
auth_oauth2.resource_server_id = rabbitmq
# Amazon Cognito does not include an audience field in access tokens
auth_oauth2.verify_aud = false

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

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

此組態使用範圍別名，將 Amazon Cognito 中定義的範圍映射至 RabbitMQ 相容範圍。

3. 使用 update-configuration AWS CLI 命令更新組態，如下列範例所示。在此命令中，新增您在此程序步驟 1 的回應中收到的組態 ID。例如 **c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca**。

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

此命令會傳回類似下列範例的回應。

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

```
    "Name": "rabbitmq-oauth2-config",
    "Warnings": []
}
```

4. 使用您在此程序的步驟 2 中建立的 OAuth 2.0 組態來建立代理程式。若要這樣做，請使用 create-broker AWS CLI 命令，如下列範例所示。在此命令中，分別提供您在步驟 1 和 2 回應中取得的組態 ID 和修訂編號。例如，**c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca** 和 **2**。

```
aws mq create-broker \
 --broker-name "rabbitmq-oauth2-broker" \
 --engine-type "RABBITMQ" \
 --engine-version "3.13" \
 --host-instance-type "mq.m7g.large" \
 --deployment-mode "CLUSTER_MULTI_AZ" \
 --logs '{"General": true}' \
 --publicly-accessible \
 --configuration '{"Id": "<c-fa3390a5-7e01-4559-ae0c-
eb15b38b22ca>","Revision": <2>}' \
```

此命令會傳回類似下列範例的回應。

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

5. 使用 describe-broker AWS CLI 命令，確認代理程式的狀態從 轉換為 CREATION_IN_PROGRESS RUNNING，如下列範例所示。在此命令中，提供您在上一個步驟的結果中取得的代理程式 ID 例如，**b-2a1b5133-a10c-49d2-879b-8c176c34cf73**。

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

此命令會傳回類似下列範例的回應。下列回應是 describe-broker命令傳回的完整輸出的縮寫版本。此回應顯示代理程式狀態和用於保護代理程式的身分驗證策略。在此情況下，config_managed身分驗證策略表示代理程式使用 OAuth 2 身分驗證方法。

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

```
    "BrokerState": "RUNNING",
    ...
}
```

若要使用 OAuth2 登入 RabbitMQ 管理主控台，代理程式端點需要在對應的 Amazon Cognito 應用程式用戶端中新增為有效的回呼 URL。如需詳細資訊，請參閱 Amazon Cognito CDK 堆疊範例設定中的步驟 5。

6. 使用以下`perf-test.sh`指令碼驗證 OAuth 2.0 身分驗證和授權。

使用此 bash 指令碼來測試 Amazon MQ for RabbitMQ 代理程式的連線能力。此指令碼會從 Amazon Cognito 取得權杖，並驗證連線是否已正確設定。如果設定成功，您會看到代理程式發佈並取用訊息。

如果收到`ACCESS_REFUSED`錯誤，您可以使用代理程式的 CloudWatch 日誌對組態設定進行疑難排解。您可以在 Amazon MQ 主控台中找到代理程式的 CloudWatch 日誌群組連結。

在此指令碼中，您需要提供下列值：

- `CLIENT_ID` 和 `CLIENT_SECRET`：您可以在 Amazon Cognito 主控台的應用程式用戶端頁面上找到這些值。
- Cognito 網域：您可以在 Amazon Cognito 主控台中找到此項目。在品牌下，選擇網域。在網域頁面上，您可以在資源伺服器區段下找到此值。
- Amazon MQ 代理程式端點：您可以在 Amazon MQ 主控台代理程式詳細資訊頁面上的連線下找到此值。

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

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

# FIXME: Update this value with the domain of your Amazon Cognito user pool
RESPONSE=$(curl -X POST ${RabbitMqOAuth2TestStack.TokenEndpoint} \
                -H "Content-Type: application/x-www-form-urlencoded" \
```

```
                -d
 "grant_type=client_credentials&client_id=${CLIENT_ID}&client_secret=${CLIENT_SECRET}&scope
configure:all rabbitmq/read:all rabbitmq/tag:administrator rabbitmq/write:all")


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

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

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

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

使用 Amazon Cognito 設定 OAuth 2.0 和簡易身分驗證

當您使用 OAuth 2.0 身分驗證建立代理程式時，您可以指定下列其中一種身分驗證方法：

- 僅限 OAuth 2.0：若要使用此方法，請勿在建立代理程式時提供使用者名稱和密碼。先前的程序說明如何僅使用 OAuth 2.0 身分驗證方法。
- OAuth 2.0 和簡單身分驗證：若要使用此方法，請在建立代理程式時提供使用者名稱和密碼。此外，請將 `auth_backends.2 = internal`新增至您的代理程式組態，如下列程序所示。

在下列程序中，請務必以其實際值取代預留位置值，例如 *<ConfigurationId>* 和 *<Revision>*。

1. 若要使用這兩種身分驗證方法，請建立您的代理程式組態，如下列範例所示。

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

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

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

   此組態使用範圍別名，將 Amazon Cognito 中定義的範圍映射至 RabbitMQ 相容範圍。

2. 建立同時使用身分驗證方法的代理程式，如下列範例所示。

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

3. 驗證代理程式狀態和設定身分驗證方法的組態是否成功，如使用 Amazon Cognito 設定 OAuth 2.0 身分驗證程序的步驟 5 和 6 所述。

使用 Amazon MQ for RabbitMQ 的 IAM 身分驗證和授權

下列程序示範如何啟用 AWS Amazon MQ for RabbitMQ 代理程式的 IAM 身分驗證和授權。啟用 IAM 後，使用者可以使用 AWS IAM 憑證進行身分驗證，以存取 RabbitMQ Management API 並透過

AMQP 連線。如需 IAM 身分驗證如何與 Amazon MQ for RabbitMQ 搭配使用的詳細資訊，請參閱 the section called “IAM 身分驗證和授權”。

先決條件

- AWS 擁有 Amazon MQ for RabbitMQ 代理程式 AWS 之帳戶的管理員登入資料
- 使用這些管理員登入資料設定的 shell 環境 （使用 CLI AWS 設定檔或環境變數）
- AWS 已安裝和設定的 CLI
- `jq` 已安裝命令列 JSON 處理器
- `curl` 已安裝命令列工具

使用 設定 IAM 身分驗證和授權 AWS CLI

1. 設定環境變數

   為您的代理程式設定所需的環境變數：

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

2. 啟用傳出 JWT 字符

   為 AWS 您的帳戶啟用傳出 Web 聯合身分：

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

   輸出會以 格式顯示您帳戶的唯一發行者識別符 URL`https://<id>.tokens.sts.global.api.aws`。

3. 建立 IAM 政策文件

   建立政策文件，授予取得 Web 身分字符的許可：

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

4. 建立信任政策

   擷取您的發起人身分並建立信任政策文件：

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

5. 建立 IAM 角色

   建立 IAM 角色並連接政策：

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

6. 設定 RabbitMQ OAuth2 設定

   使用 OAuth2 身分驗證和授權設定建立 RabbitMQ 組態檔案：

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

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

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

management.oauth_enabled = true
EOF
```

7. 更新代理程式組態

   將新組態套用至您的代理程式：

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

# Create a new configuration revision
```

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

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

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

等待代理程式狀態回到 ，RUNNING再繼續下一個步驟。

8. 取得 JWT 字符

擔任 IAM 角色並取得 Web 身分字符：

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

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

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

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

9. 存取 RabbitMQ Management API

使用 JWT 字符存取 RabbitMQ 管理 API：

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

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

成功的回應會確認 IAM 身分驗證是否正常運作。回應包含 JSON 格式的代理程式概觀資訊。

10. 使用 JWT 字符透過 AMQP 連接

    使用 JWT 字符搭配 perf-test 工具測試 AMQP 連線：

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

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

如果您收到ACCESS_REFUSED錯誤，您可以使用代理程式的 CloudWatch 日誌對組態設定進行疑難排解。您可以在 Amazon MQ 主控台中找到代理程式的 CloudWatch Logs 日誌群組連結。

使用 Amazon MQ for RabbitMQ 的 LDAP 身分驗證和授權

本教學說明如何使用 設定 Amazon MQ for RabbitMQ 代理程式的 LDAP 身分驗證和授權 AWS Managed Microsoft AD。

在此頁面

設定 LDAP 身分驗證和授權的先決條件

您可以部署 AWS Amazon MQ for RabbitMQ LDAP 整合的 CDK 堆疊 AWS Managed Microsoft AD，藉此設定本教學課程中所需的 AWS 資源。

此 CDK 堆疊會自動建立所有必要 AWS 的資源 AWS Managed Microsoft AD，包括 LDAP 使用者和群組、Network Load Balancer、憑證和 IAM 角色。如需堆疊所建立資源的完整清單，請參閱套件 README。

如果您要手動設定資源，而不是使用 CDK 堆疊，請確保在 Amazon MQ for RabbitMQ 代理程式上設定 LDAP 之前，已具備同等的基礎設施。

設定 Amazon MQ 的先決條件

AWS CLI 版本 >= 2.28.23，以便在建立代理程式期間選擇性新增使用者名稱和密碼。

使用 CLI 在 RabbitMQ AWS 中設定 LDAP

此程序使用 AWS CLI 來建立和設定必要的資源。在下列程序中，請務必使用其實際值取代預留位置值，例如 configurationID 和 Revision<2>，<c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca>以及 。

1. 使用 `create-configuration` AWS CLI 命令建立新的組態，如下列範例所示。

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

此命令會傳回類似下列範例的回應。

```
{
"Arn": "arn:aws:mq:us-west-2:123456789012:configuration:c-fa3390a5-7e01-4559-ae0c-
eb15b38b22ca",
    "AuthenticationStrategy": "simple",
    "Created": "2025-07-17T16:03:01.759943+00:00",
    "Id": "c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca",
    "LatestRevision": {
"Created": "2025-07-17T16:03:01.759000+00:00",
```

```
    "Description": "Auto-generated default for rabbitmq-ldap-config on RabbitMQ
 3.13",
    "Revision": 1
    },
    "Name": "rabbitmq-ldap-config"
}
```

2. 建立名為 的組態檔案rabbitmq.conf，以使用 LDAP 做為身分驗證和授權方法，如下列範例所示。將範本中的所有預留位置值（以 標記${RabbitMqLdapTestStack.*}) 取代為您部署 AWS CDK 的先決條件堆疊輸出或同等基礎設施的實際值。

```
auth_backends.1 = ldap

# LDAP authentication settings - For more information,
# see https://www.rabbitmq.com/docs/ldap#basic

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

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

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

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

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

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

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

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

3. 使用 update-configuration AWS CLI 命令更新組態，如下列範例所示。在此命令中，新增您在此程序步驟 1 的回應中收到的組態 ID。例如 c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca。

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

此命令會傳回類似下列範例的回應。

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

4. 使用您在此程序的步驟 2 中建立的 LDAP 組態來建立代理程式。若要這樣做，請使用 create-broker AWS CLI 命令，如下列範例所示。在此命令中，分別提供您在步驟 1 和 2 回應中取得的組態 ID 和修訂編號。例如，c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca 和 2。

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

此命令會傳回類似下列範例的回應。

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

⚠ 中介裝置命名限制

先決條件 CDK 堆疊建立的 IAM 角色會將代理程式名稱限制為以 開頭`rabbitmq-ldap-test`。確保您的代理程式名稱遵循此模式，否則 IAM 角色將無法擔任 ARN 解析的角色。

5. 使用 `describe-broker` AWS CLI 命令，確認代理程式的狀態從 轉換為 `CREATION_IN_PROGRESS` `RUNNING`，如下列範例所示。在此命令中，提供您在上一個步驟的結果中取得的代理程式 ID 例如，`b-2a1b5133-a10c-49d2-879b-8c176c34cf73`。

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

此命令會傳回類似下列範例的回應。下列回應是 `describe-broker`命令傳回的完整輸出的縮寫版本。此回應顯示代理程式狀態和用於保護代理程式的身分驗證策略。在此情況下，`config_managed`身分驗證策略表示代理程式使用 LDAP 身分驗證方法。

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

6. 使用先決條件 CDK 堆疊建立的其中一個測試使用者來驗證 RabbitMQ 存取

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

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

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

使用 Amazon MQ for RabbitMQ 的 HTTP 身分驗證和授權

本教學說明如何使用外部 HTTP 伺服器設定 Amazon MQ for RabbitMQ 代理程式的 HTTP 身分驗證和授權。

> **Note**
>
> HTTP 身分驗證外掛程式僅適用於 Amazon MQ for RabbitMQ 第 4 版及更新版本。

在此頁面

設定 HTTP 身分驗證和授權的先決條件

您可以部署 AWS Amazon MQ for RabbitMQ HTTP 身分驗證整合的 CDK 堆疊，藉此設定本教學課程中所需的 AWS 資源。

此 CDK 堆疊會自動建立所有必要 AWS 的資源，包括 HTTP 身分驗證伺服器、憑證和 IAM 角色。如需堆疊所建立資源的完整清單，請參閱套件 README。

如果您要手動設定資源，而不是使用 CDK 堆疊，請確保在 Amazon MQ for RabbitMQ 代理程式上設定 HTTP 身分驗證之前，已具備同等的基礎設施。

設定 Amazon MQ 的先決條件

AWS CLI 版本 >= 2.28.23，以便在建立代理程式期間選擇性新增使用者名稱和密碼。

使用 CLI 在 RabbitMQ 中設定 HTTP AWS 身分驗證

此程序使用 AWS CLI 來建立和設定必要的資源。在下列程序中，請務必將預留位置值取代為其實際值。

1. 使用 `create-configuration` AWS CLI 命令建立新的組態，如下列範例所示。

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

此命令會傳回類似下列範例的回應。

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

2. 建立名為 的組態檔案rabbitmq.conf，以使用 HTTP 做為身分驗證和授權方法，如下列範例所示。將範本中的所有預留位置值 （以 標示${...}) 取代為您部署 AWS CDK 的先決條件堆疊輸出或同等基礎設施的實際值。

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

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

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

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

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

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

3. 使用 update-configuration AWS CLI 命令更新組態。使用步驟 3 中的組態 ID。

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

此命令會傳回類似下列範例的回應。

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

4. 使用 HTTP 組態建立代理程式。使用先前步驟中的組態 ID 和修訂編號。

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

此命令會傳回類似下列範例的回應。

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

5. 使用 describe-broker AWS CLI CREATION_IN_PROGRESS 命令RUNNING，確認代理程式的狀態從 轉換為 。

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

此命令會傳回類似下列範例的回應。config_managed 身分驗證策略表示代理程式使用 HTTP 身分驗證方法。

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

6. 使用先決條件 CDK 堆疊建立的其中一個測試使用者來驗證 RabbitMQ 存取

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

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

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

針對 Amazon MQ for RabbitMQ 使用 SSL 憑證身分驗證

本教學課程說明如何使用私有憑證授權機構設定 Amazon MQ for RabbitMQ 代理程式的 SSL 憑證驗證。

> **Note**
>
> SSL 憑證身分驗證外掛程式僅適用於 Amazon MQ for RabbitMQ 第 4 版及更高版本。

在此頁面

設定 SSL 憑證身分驗證的先決條件

SSL 憑證驗證使用交互 TLS (mTLS) 來驗證使用 X.509 憑證的用戶端。您可以部署 AWS Amazon MQ for RabbitMQ mTLS 整合的 CDK 堆疊，藉此設定本教學課程中所需的 AWS 資源。

此 CDK 堆疊會自動建立所有必要 AWS 的資源，包括憑證授權單位、用戶端憑證和 IAM 角色。如需堆疊所建立資源的完整清單，請參閱套件 README。

> **Note**
>
> 部署 CDK 堆疊之前，請設定 `RABBITMQ_TEST_USER_NAME`環境變數。此值將用作用戶端憑證中的通用名稱 (CN)，並且必須符合您在教學課程中使用的使用者名稱。例如：`export RABBITMQ_TEST_USER_NAME="myuser"`

如果您要手動設定資源，而不是使用 CDK 堆疊，請確保在 Amazon MQ for RabbitMQ 代理程式上設定 SSL 憑證身分驗證之前，已具備同等的基礎設施。

設定 Amazon MQ 的先決條件

AWS CLI 版本 >= 2.28.23，以便在建立代理程式期間選擇性新增使用者名稱和密碼。

使用 CLI 在 RabbitMQ 中設定 SSL AWS 憑證身分驗證

此程序使用 AWS CLI 來建立和設定必要的資源。在下列程序中，請務必使用其實際值取代預留位置值，例如 configurationID 和 Revision<2>，<c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca>以及 。

1. 使用 `create-configuration` AWS CLI 命令建立新的組態，如下列範例所示。

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

此命令會傳回類似下列範例的回應。

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

2. 建立名為 的組態檔案`rabbitmq.conf`，以使用 SSL 憑證身分驗證，如下列範例所示。將範本中的所有預留位置值 （ 以 標示`${...}`) 取代為您部署 AWS CDK 的先決條件堆疊輸出或同等基礎設施的實際值。

```
auth_mechanisms.1 = EXTERNAL
ssl_cert_login_from = common_name
```

```
auth_backends.1 = internal

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

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

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

3. 使用 update-configuration AWS CLI 命令更新組態，如下列範例所示。在此命令中，新增您在此程序步驟 1 的回應中收到的組態 ID。例如 c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca。

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

此命令會傳回類似下列範例的回應。

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

4. 使用您在此程序的步驟 2 中建立的 SSL 憑證身分驗證組態來建立代理程式。若要這樣做，請使用 create-broker AWS CLI 命令，如下列範例所示。在此命令中，分別提供您在步驟 1 和 2 回應中取得的組態 ID 和修訂編號。例如，c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca 和 2。

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

此命令會傳回類似下列範例的回應。

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

5. 使用 describe-broker AWS CLI 命令，確認代理程式的狀態從 轉換為 CREATION_IN_PROGRESS RUNNING，如下列範例所示。在此命令中，提供您在上一個步驟的結果中取得的代理程式 ID。例如 b-2a1b5133-a10c-49d2-879b-8c176c34cf73。

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

此命令會傳回類似下列範例的回應。下列回應是 describe-broker命令傳回的完整輸出的縮寫版本。此回應顯示代理程式狀態和用於保護代理程式的身分驗證策略。在此情況下，config_managed身分驗證策略表示代理程式使用 SSL 憑證身分驗證方法。

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

6. 使用以下ssl.sh指令碼驗證 SSL 憑證身分驗證。

使用此 bash 指令碼來測試 Amazon MQ for RabbitMQ 代理程式的連線能力。此指令碼使用您的用戶端憑證進行身分驗證，並驗證連線是否已正確設定。如果設定成功，您會看到代理程式發佈並取用訊息。

如果您收到ACCESS_REFUSED錯誤，您可以使用代理程式的 CloudWatch 日誌對組態設定進行疑難排解。您可以在 Amazon MQ 主控台中找到代理程式的 CloudWatch 日誌群組連結。

在此指令碼中，您需要提供下列值：

- USERNAME：用戶端憑證的通用名稱 (CN)。
- CLIENT_KEYSTORE：用戶端金鑰存放區檔案的路徑 (PKCS12 格式) 。如果您使用先決條件 CDK 堆疊，預設路徑為 $(pwd)/certs/client-keystore.p12。
- KEYSTORE_PASSWORD：用戶端金鑰存放區的密碼。如果您使用先決條件 CDK 堆疊，預設密碼為 changeit。
- BROKER_DNS：您可以在 Amazon MQ 主控台的代理程式詳細資訊頁面的連線下找到此值。

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

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

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

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

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

將 mTLS 用於 AMQP 和管理端點

本教學課程說明如何使用私有憑證授權機構為 AMQP 用戶端連線和 RabbitMQ 管理界面設定交互 TLS (mTLS)。

> **Note**
>
> 針對 mTLS 使用私有憑證授權單位僅適用於 Amazon MQ for RabbitMQ 第 4 版及更高版本。

在此頁面

設定 mTLS 的先決條件

您可以部署 AWS Amazon MQ for RabbitMQ mTLS 與 整合的 CDK 堆疊，藉此設定本教學課程中所需的 AWS 資源。

此 CDK 堆疊會自動建立所有必要 AWS 的資源，包括憑證授權單位、用戶端憑證和 IAM 角色。如需堆疊所建立資源的完整清單，請參閱套件 README。

如果您要手動設定資源，而不是使用 CDK 堆疊，請確保您在 Amazon MQ for RabbitMQ 代理程式上設定 mTLS 之前已具備同等的基礎設施。

設定 Amazon MQ 的先決條件

AWS CLI 版本 >= 2.28.23，以便在建立代理程式期間選擇性新增使用者名稱和密碼。

使用 CLI 在 RabbitMQ AWS 中設定 mTLS

此程序使用 AWS CLI 來建立和設定必要的資源。在下列程序中，請務必使用其實際值取代預留位置值，例如 configurationID 和 Revision<2>，<c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca>以及 。

1. 使用 `create-configuration` AWS CLI 命令建立新的組態，如下列範例所示。

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

此命令會傳回類似下列範例的回應。

```
{
    "Arn": "arn:aws:mq:us-west-2:123456789012:configuration:c-fa3390a5-7e01-4559-
ae0c-eb15b38b22ca",
    "AuthenticationStrategy": "simple",
    "Created": "2025-07-17T16:03:01.759943+00:00",
    "Id": "c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca",
    "LatestRevision": {
        "Created": "2025-07-17T16:03:01.759000+00:00",
```

```
        "Description": "Auto-generated default for rabbitmq-mtls-config on RabbitMQ
 4.2",
        "Revision": 1
    },
    "Name": "rabbitmq-mtls-config"
}
```

2. 建立名為 的組態檔案`rabbitmq.conf`，以設定 AMQP 和管理端點的 mTLS，如下列範例所示。將範本中的所有預留位置值 （以 標示`${...}`) 取代為您部署 AWS CDK 的先決條件堆疊輸出或同等基礎設施的實際值。

```
auth_backends.1 = internal

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

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

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

3. 使用 `update-configuration` AWS CLI 命令更新組態，如下列範例所示。在此命令中，新增您在此程序步驟 1 的回應中收到的組態 ID。例如 `c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca`。

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

此命令會傳回類似下列範例的回應。

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

4. 使用您在此程序的步驟 2 中建立的 mTLS 組態來建立代理程式。若要這樣做，請使用 create-broker AWS CLI 命令，如下列範例所示。在此命令中，分別提供您在步驟 1 和 2 回應中取得的組態 ID 和修訂編號。例如，c-fa3390a5-7e01-4559-ae0c-eb15b38b22ca 和 2。

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

此命令會傳回類似下列範例的回應。

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

5. 使用 describe-broker AWS CLI 命令，確認代理程式的狀態從 轉換為 CREATION_IN_PROGRESS RUNNING，如下列範例所示。在此命令中，提供您在上一個步驟的結果中取得的代理程式 ID。例如 b-2a1b5133-a10c-49d2-879b-8c176c34cf73。

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

此命令會傳回類似下列範例的回應。下列回應是 describe-broker命令傳回的完整輸出的縮寫版本。

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

6. 使用以下mtls.sh指令碼驗證 mTLS 身分驗證。

使用此 bash 指令碼來測試 Amazon MQ for RabbitMQ 代理程式的連線能力。此指令碼使用您的用戶端憑證來驗證和驗證連線是否已正確設定。如果設定成功，您會看到代理程式發佈並取用訊息。

如果您收到ACCESS_REFUSED錯誤，您可以使用代理程式的 CloudWatch 日誌對組態設定進行疑難排解。您可以在 Amazon MQ 主控台中找到代理程式的 CloudWatch 日誌群組連結。

在此指令碼中，您需要提供下列值：

- USERNAME 和 PASSWORD：您使用代理程式建立的 RabbitMQ 使用者登入資料。
- CLIENT_KEYSTORE：用戶端金鑰存放區檔案的路徑 (PKCS12 格式）。如果您使用先決條件 CDK 堆疊，預設路徑為 $(pwd)/certs/client-keystore.p12。
- KEYSTORE_PASSWORD：用戶端金鑰存放區的密碼。如果您使用先決條件 CDK 堆疊，預設密碼為 changeit。
- BROKER_DNS：您可以在 Amazon MQ 主控台的代理程式詳細資訊頁面的連線下找到此值。

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

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

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

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

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

連接您的 JMS 應用程式

本教學課程說明如何使用 RabbitMQ JMS 用戶端將 JMS 應用程式連線至 Amazon MQ for RabbitMQ 代理程式。您將了解如何建立生產者以傳送訊息，以及建立取用者以接收來自 RabbitMQ 佇列的訊息。

開始之前，請將適當的 RabbitMQ JMS 相依性新增至 Maven 專案：

對於 JMS 1.1 和 2.0：

```
<dependencies>

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

</dependencies>
```

對於 JMS 3.1：

```
<dependencies>

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

</dependencies>
```

建立生產者

下列程式碼範例示範如何使用 JMS 寫入 RabbitMQ 佇列：

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

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

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

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

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

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

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

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

建立取用者

下列程式碼範例示範如何使用 JMS 從 RabbitMQ 佇列讀取：

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

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

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

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

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

```
destination.setAmqpQueueName(queueName);

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

connection.start();
```

Amazon MQ 的安全性

的雲端安全性 AWS 是最高優先順序。身為 AWS 客戶，您可以受益於資料中心和網路架構，這些架構是為了滿足最安全敏感組織的需求而建置。

安全性是 AWS 與您之間的共同責任。共同責任模式將其描述為雲端的安全性，和雲端中的安全性：

- 雲端的安全性 – AWS 負責保護在 AWS Cloud 中執行 AWS 服務的基礎設施。 AWS 也為您提供可安全使用的服務。在AWS 合規計畫中，第三方稽核人員會定期測試和驗證我們安全的有效性。若要了解適用於 Amazon MQ 的合規計劃，請參閱AWS 合規計劃的 服務範圍。
- 雲端的安全性 – 您的責任取決於您使用 AWS 的服務。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。您也會了解如何使用其他 AWS 服務來協助您監控和保護 Amazon MQ 資源。

主題

Amazon MQ 的資料保護

共同責任模型適用於 AWS Amazon MQ 中的資料保護。如此模型所述， AWS 負責保護執行所有 的全域基礎設施 AWS 雲端。您負責維護在此基礎設施上託管內容的控制權。您也同時負責所使用 AWS 服務 的安全組態和管理任務。如需資料隱私權的詳細資訊，請參閱資料隱私權常見問答集。 如需歐洲資料保護的詳細資訊，請參閱一般資料保護規則 (GDPR) 中心。

基於資料保護目的，我們建議您保護 AWS 帳戶 登入資料，並使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 設定個別使用者。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 設定 API 和使用者活動記錄 AWS CloudTrail。如需有關使用 CloudTrail 追蹤擷取 AWS 活動的資訊，請參閱AWS CloudTrail 《 使用者指南》中的使用 CloudTrail 追蹤。
- 使用 AWS 加密解決方案，以及其中的所有預設安全控制 AWS 服務。
- 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 AWS 透過命令列界面或 API 存取 時需要 FIPS 140-3 驗證的密碼編譯模組，請使用 FIPS 端點。如需有關 FIPS 和 FIPS 端點的更多相關資訊，請參閱聯邦資訊處理標準 (FIPS) 140-3。

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 或使用主控台、API AWS CLI或 AWS SDKs的其他 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。其他 AWS 服務可存取中介裝置名稱和使用者名稱，包括 CloudWatch Logs。代理程式使用者名稱不適用於私有或敏感資料。

> **Important**
>
> TLS 1.3 不適用於 RabbitMQ 代理程式。

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 AWS KMS key 您希望 Amazon MQ 用來加密靜態資料的 。如果您未指定 KMS 金鑰，Amazon MQ 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。刪除代理程式以撤銷存取權。
> - 對於使用 Amazon Elastic File System (EFS) 存放訊息資料的 Amazon MQ for ActiveMQ 代理程式，在採取必要動作後，可能需要數小時才能撤銷您帳戶中的 KMS 金鑰。 Amazon Elastic File System EFS
> - 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。
> - 在採取必要的動作之後，可能需要數小時才能停用金鑰或撤銷授予。
> - 對於加密或解密 CloudWatch 日誌，您無法設定 Amazon MQ 用於加密金鑰的內容。CloudWatch 日誌使用加密保護靜態資料，並加密日誌群組。CloudWatch 記錄服務會預設地管理伺服器端加密。如需日誌群組加密方式的詳細資訊，請參閱 Amazon CloudWatch Logs 使用者指南。

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，而且 AWS 資料中心之間的所有網路流量都會在實體層透明加密。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 AWS 服務 ，可協助管理員安全地控制對 AWS 資源的存取。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 是您可以免費使用 AWS 服務 的 。

主題

- 目標對象
- 使用身分驗證

- 使用政策管理存取權
- Amazon MQ 如何搭配 IAM 運作
- Amazon MQ 身分型政策範例
- Amazon MQ 的 API 身分驗證和授權
- 中介裝置身分驗證和授權
- AWS Amazon MQ 的 受管政策
- 使用 Amazon MQ 的服務連結角色
- Amazon MQ 身分識別和存取疑難排解

目標對象

使用方式 AWS Identity and Access Management (IAM) 會根據您的角色而有所不同：

- 服務使用者 — 若無法存取某些功能，請向管理員申請所需許可 (請參閱 Amazon MQ 身分識別和存取疑難排解)
- 服務管理員 — 負責設定使用者存取權並提交相關許可請求 (請參閱 Amazon MQ 如何搭配 IAM 運作)
- IAM 管理員 — 撰寫政策以管理存取控制 (請參閱 Amazon MQ 身分型政策範例)

使用身分驗證

身分驗證是您 AWS 使用身分憑證登入 的方式。您必須以 AWS 帳戶根使用者、IAM 使用者或擔任 IAM 角色身分進行身分驗證。

您可以使用身分來源的登入資料，例如 AWS IAM Identity Center (IAM Identity Center)、單一登入身分驗證或 Google/Facebook 登入資料，以聯合身分的形式登入。如需有關登入的詳細資訊，請參閱《AWS 登入 使用者指南》中的如何登入您的 AWS 帳戶。

對於程式設計存取， AWS 提供 SDK 和 CLI 以密碼編譯方式簽署請求。如需詳細資訊，請參閱《IAM 使用者指南》中的 API 請求的AWS 第 4 版簽署程序。

AWS 帳戶 根使用者

當您建立 時 AWS 帳戶，您會從一個名為 AWS 帳戶 theroot 使用者的登入身分開始，該身分可完整存取所有 AWS 服務 和 資源。強烈建議不要使用根使用者來執行日常任務。有關需要根使用者憑證的任務，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https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users.html是一種身分具備單人或應用程式的特定許可權。建議以臨時憑證取代具備長期憑證的 IAM 使用者。如需詳細資訊，請參閱《IAM 使用者指南》中的要求人類使用者使用聯合身分提供者來 AWS 使用臨時憑證存取 。

IAM 群組會指定 IAM 使用者集合，使管理大量使用者的許可權更加輕鬆。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM 使用者的使用案例。

IAM 角色

IAM 角色https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html的身分具有特定許可權，其可以提供臨時憑證。您可以透過從使用者切換到 IAM 角色 （主控台） 或呼叫 AWS CLI 或 AWS API 操作來擔任角色。如需詳細資訊，請參閱《IAM 使用者指南》中的擔任角色的方法。

IAM 角色適用於聯合身分使用者存取、臨時 IAM 使用者許可、跨帳戶存取權與跨服務存取，以及在 Amazon EC2 執行的應用程式。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM 中的快帳戶資源存取。

使用政策管理存取權

您可以透過建立政策並將其連接到身分或資源 AWS 來控制 AWS 中的存取。政策定義與身分或資源相關聯的許可。當委託人提出請求時 AWS ， 會評估這些政策。大多數政策會以 JSON 文件 AWS 形式存放在 中。如需進一步了解 JSON 政策文件，請參閱《IAM 使用者指南》中的 JSON 政策概觀。

管理員會使用政策，透過定義哪些主體可在哪些條件下對哪些資源執行動作，以指定可存取的範圍。

預設情況下，使用者和角色沒有許可。IAM 管理員會建立 IAM 政策並將其新增至角色，供使用者後續擔任。IAM 政策定義動作的許可，無論採用何種方式執行。

身分型政策

身分型政策是附加至身分 (使用者、使用者群組或角色) 的 JSON 許可政策文件。這類政策控制身分可對哪些資源執行哪些動作，以及適用的條件。如需了解如何建立身分型政策，請參閱《IAM 使用者指南》中的透過客戶管理政策定義自訂 IAM 許可。

身分型政策可分為內嵌政策 (直接內嵌於單一身分) 與受管政策 (可附加至多個身分的獨立政策)。如需了解如何在受管政策及內嵌政策之間做選擇，請參閱《IAM 使用者指南》中的在受管政策與內嵌政策之間選擇。

資源型政策

資源型政策是附加到資源的 JSON 政策文件。範例包括 IAM 角色信任政策與 Amazon S3 儲存貯體政策。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。您必須在資源型政策中指定主體。

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 S3 AWS WAF和 Amazon VPC 是支援 ACLs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 支援其他政策類型，可設定更多常見政策類型授予的最大許可：

- 許可界限 — 設定身分型政策可授與 IAM 實體的最大許可。如需詳細資訊，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) — 為 AWS Organizations中的組織或組織單位指定最大許可。如需詳細資訊，請參閱《AWS Organizations 使用者指南》中的服務控制政策。
- 資源控制政策 (RCP) — 設定您帳戶中資源可用許可的上限。如需詳細資訊，請參閱《AWS Organizations 使用者指南》中的資源控制政策 (RCP)。
- 工作階段政策 — 在以程式設計方式為角色或聯合身分使用者建立臨時工作階段時，以參數形式傳遞的進階政策。如需詳細資訊，請參《IAM 使用者指南》中的工作階段政策。

多種政策類型

當多種類型的政策適用於請求時，產生的許可會更複雜而無法理解。若要了解如何 AWS 在涉及多種政策類型時決定是否允許請求，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。若要全面了解 Amazon MQ 和其他 AWS 服務如何與 IAM 搭配使用，請參閱《IAM 使用者指南》中的與 IAM AWS 搭配使用的 服務。

Amazon MQ 使用 IAM for Amazon MQ API 操作來建立、更新、刪除和列出代理程式。對於發佈和訂閱訊息的代理程式存取權，Amazon MQ for ActiveMQ 支援原生 ActiveMQ 身分驗證和 LDAP，而 Amazon MQ for RabbitMQ 支援 IAM 身分驗證和其他方法。如需詳細資訊，請參閱the section called “中介裝置身分驗證和授權”。

主題

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。政策會使用動作來授予執行相關聯動作的許可。

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。

Resources

管理員可以使用 AWS JSON 政策來指定誰可以存取內容。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。最佳實務是使用其 Amazon Resource Name (ARN) 來指定資源。若動作不支援資源層級許可，使用萬用字元 (*) 表示該陳述式適用於所有資源。

```
"Resource": "*"
```

在 Amazon MQ 中，主要 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	條件金鑰
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 ARNs 格式的詳細資訊，請參閱 Amazon Resource Name (ARNs AWS 和服務命名空間。

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。

條件索引鍵

管理員可以使用 AWS JSON 政策來指定誰可以存取內容。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` 元素會根據定義的條件，指定陳述式的執行時機。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。若要查看所有 AWS 全域條件索引鍵，請參閱《IAM 使用者指南》中的AWS 全域條件內容索引鍵。

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。

條件金鑰	描述	Type
aws:RequestTag/${TagKey}	根據要求中傳遞的標籤篩選動作。	String
aws:ResourceTag/${TagKey}	根據與資源相關聯的標籤篩選動作。	String

條件金鑰	描述	Type
aws:TagKeys	根據要求中傳遞的標籤鍵篩選動作。	String

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。如需根據標籤控制存取，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。

JSON

```
{
    "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：

- 將標籤新增至 Amazon MQ 資源
- 使用 IAM 標籤控制存取

Amazon MQ IAM 角色

IAM 角色是您 AWS 帳戶中具有特定許可的實體。

搭配使用暫時登入資料與 Amazon MQ

您可以搭配聯合使用暫時憑證、擔任 IAM 角色，或是擔任跨帳戶角色。您可以透過呼叫 AssumeRole 或 GetFederationToken 等 AWS STS API 操作來取得臨時安全登入資料。

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 管理主控台 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 作業的所需許可。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 受管政策並邁向最低權限許可 – 若要開始將許可授予您的使用者和工作負載，請使用將許可授予許多常見使用案例的 AWS 受管政策。它們可在您的 中使用 AWS 帳戶。我們建議您定義特定於使用案例 AWS 的客戶受管政策，以進一步減少許可。如需更多資訊，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 套用許可的更多相關資訊，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。如果透過特定 例如 使用服務動作 AWS 服務，您也可以使用條件來授予其存取權 CloudFormation。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，可協助您撰寫安全且實用的政策。如需詳細資訊，請參閱《IAM 使用者指南》中的使用 IAM Access Analyzer 驗證政策。
- 需要多重要素驗證 (MFA) – 如果您的案例需要 IAM 使用者或 中的根使用者 AWS 帳戶，請開啟 MFA 以提高安全性。如需在呼叫 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》中的透過 MFA 的安全 API 存取。

如需 IAM 中最佳實務的相關資訊，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 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。

為了確保這些實體仍然可以使用 Amazon MQ 主控台，請將下列 AWS 受管政策連接至實體。如需更多資訊，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，您不需要允許最低主控台許可。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 使用者檢視附加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或使用 或 AWS CLI AWS API 以程式設計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 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 其他許可參考
- 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。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Action": [
            "mq:*",
            "ec2:CreateNetworkInterface",
            "ec2:DeleteNetworkInterface",
            "ec2:DetachNetworkInterface",
            "ec2:DescribeInternetGateways",
            "ec2:DescribeNetworkInterfaces",
            "ec2:DescribeRouteTables",
            "ec2:DescribeSecurityGroups",
            "ec2:DescribeSubnets",
            "ec2:DescribeVpcs"
        ],
        "Effect": "Allow",
        "Resource": "*"
    },{
        "Action": [
            "ec2:CreateNetworkInterfacePermission",
            "ec2:DeleteNetworkInterfacePermission",
            "ec2:DescribeNetworkInterfacePermissions"
        ],
        "Effect": "Allow",
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "ec2:AuthorizedService": "mq.amazonaws.com"
            }
        }
    }]
}
```

如需詳細資訊，請參閱Setting Up Amazon MQ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 其他許可參考

下表列出 Amazon MQ API 和特定功能所需的其他 IAM 許可，例如 OAuth 2.0 身分驗證。

Amazon MQ REST API	權限	說明
UpdateBroker	`mq:UpdateBrokerAccessConfiguration`	您需要此許可，才能更新相關聯代理程式組態中的身分驗證和授權選項。如需詳細資訊，請參閱Amazon MQ for RabbitMQ 的 OAuth 2.0 身分驗證和授權。

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*

API 動作	資源類型 (*必填項目)
DeleteUser	中介裝置*
DescribeBroker	中介裝置*
DescribeConfiguration	組態*
DescribeConfigurationRevision	組態*
DescribeUser	中介裝置*
ListConfigurationRevisions	組態*
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*

中介裝置身分驗證和授權

Amazon MQ 根據您的代理程式引擎類型，提供不同的身分驗證和授權方法。

Amazon MQ for ActiveMQ 的身分驗證和授權

Amazon MQ for ActiveMQ 支援下列身分驗證和授權方法：

簡單身分驗證和授權

在此方法中，代理程式使用者是透過 Amazon MQ 主控台或 API 建立和管理。使用者可以設定特定許可來存取佇列、主題和 ActiveMQ Web 主控台。如需此方法的詳細資訊，請參閱建立 ActiveMQ 代理程式使用者。

LDAP 身分驗證和授權

在此方法中，代理程式使用者會透過儲存在 LDAP 伺服器的登入資料進行身分驗證。您可以新增、刪除和修改使用者，並透過 LDAP 伺服器將許可指派給主題和佇列，提供集中式身分驗證和授權。如需此方法的詳細資訊，請參閱將 ActiveMQ 代理程式與 LDAP 整合。

Amazon MQ for RabbitMQ 的身分驗證和授權

Amazon MQ for RabbitMQ 支援下列身分驗證和授權方法：

簡單身分驗證和授權

在此方法中，代理程式使用者存放在 RabbitMQ 代理程式內部，並透過 Web 主控台或管理 API 進行管理。vhost、交換、佇列和主題的許可會直接在 RabbitMQ 中設定。這是預設方法。如需詳細資訊，請參閱簡易身分驗證和授權。

OAuth 2.0 身分驗證和授權

在此方法中，代理程式使用者及其許可由外部 OAuth 2.0 身分提供者 (IdP) 管理。vhost、交換、佇列和主題的使用者身分驗證和資源許可是透過 OAuth 2.0 供應商的範圍系統集中。這可簡化使用者管理，並啟用與現有身分系統的整合。如需詳細資訊，請參閱 OAuth 2.0 身分驗證和授權。

IAM 身分驗證和授權

在此方法中，代理程式使用者透過 AWS IAM 傳出聯合使用 IAM 憑證進行身分驗證。IAM 登入資料用於從 AWS Security Token Service (STS) 取得 JWT 權杖，而這些 JWT 權杖可作為 OAuth 2.0 權杖進行身分驗證。此方法利用 Amazon MQ for RabbitMQ 中現有的 OAuth 2.0 支援，其中 AWS 做為 OAuth 2.0 身分提供者。使用者身分驗證由 AWS IAM 處理，而 vhost、交換、佇列和主題的資源許可則透過 RabbitMQ 中設定的 IAM 政策和範圍別名進行管理。如需詳細資訊，請參閱 IAM 身分驗證和授權。

LDAP 身分驗證和授權

在此方法中，代理程式使用者及其許可由外部 LDAP 目錄服務管理。使用者身分驗證和資源許可是透過 LDAP 伺服器集中，允許使用者使用其現有的目錄服務憑證來存取 RabbitMQ。如需詳細資訊，請參閱 LDAP 身分驗證和授權。

HTTP 身分驗證和授權

在此方法中，代理程式使用者及其許可是由外部 HTTP 伺服器管理。使用者身分驗證和資源許可是透過 HTTP 伺服器集中，允許使用者使用自己的身分驗證和授權提供者來存取 RabbitMQ。如需此方法的詳細資訊，請參閱 HTTP 身分驗證和授權。

SSL 憑證身分驗證

Amazon MQ 支援 RabbitMQ 代理程式的交互 TLS (mTLS)。SSL 身分驗證外掛程式使用來自 mTLS 連線的用戶端憑證來驗證使用者。在此方法中，代理程式使用者會使用 X.509 用戶端憑證進行身分驗證，而非使用者名稱和密碼憑證。用戶端的憑證會根據信任的憑證授權單位 (CA) 進行驗證，使用者名稱會從憑證中的欄位擷取，例如通用名稱 (CN) 或主體別名 (SAN)。此方法提供強式身分驗證，無需透過網路傳輸登入資料。如需詳細資訊，請參閱 SSL 憑證身分驗證。

> **Note**
>
> RabbitMQ 支援多個同時使用的身分驗證和授權方法。例如，您可以同時啟用 OAuth 2.0 和簡單（內部）身分驗證。如需詳細資訊，請參閱啟用 OAuth 2.0 和簡單（內部）身分驗證的 OAuth 2.0 教學課程一節，以及 RabbitMQ 存取控制文件。 OAuth

AWS Amazon MQ 的 受管政策

AWS 受管政策是由 AWS AWS 受管政策建立和管理的獨立政策旨在為許多常用案例提供許可，以便您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予特定使用案例的最低權限許可，因為這些許可可供所有 AWS 客戶使用。我們建議您定義特定於使用案例的客戶管理政策，以便進一步減少許可。

您無法變更 AWS 受管政策中定義的許可。如果 AWS 更新受 AWS 管政策中定義的許可，則更新會影響政策連接的所有委託人身分（使用者、群組和角色）。 AWS 服務 當新的 啟動或新的 API 操作可用於現有服務時， AWS 最有可能更新 AWS 受管政策。

如需詳細資訊，請參閱 IAM 使用者指南中的 AWS 受管政策。

Amazon MQ 支援下列 AWS 受管政策：

- AmazonMQApiFullAccess
- AmazonMQApiReadOnlyAccess

- AmazonMQFullAccess
- AmazonMQReadOnlyAccess
- AmazonMQServiceRolePolicy

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”。

AWS 受管政策的 Amazon MQ 更新

檢視自此服務開始追蹤 Amazon MQ AWS 受管政策更新以來的詳細資訊。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	Date
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，並包含該服務代表您呼叫其他 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，且該許可政策無法附加至其他 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 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。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeVpcEndpoints"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateVpcEndpoint"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:vpc/*",
                "arn:aws:ec2:*:*:subnet/*",
                "arn:aws:ec2:*:*:security-group/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateVpcEndpoint"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:vpc-endpoint/*"
            ],
            "Condition": {
                "StringEquals": {
```

```
                    "aws:RequestTag/AMQManaged": "true"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": "arn:aws:ec2:*:*:vpc-endpoint/*",
            "Condition": {
                "StringEquals": {
                    "ec2:CreateAction": "CreateVpcEndpoint"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DeleteVpcEndpoints"
            ],
            "Resource": "arn:aws:ec2:*:*:vpc-endpoint/*",
            "Condition": {
                "StringEquals": {
                    "ec2:ResourceTag/AMQManaged": "true"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents",
                "logs:DescribeLogStreams",
                "logs:DescribeLogGroups",
                "logs:CreateLogStream",
                "logs:CreateLogGroup"
            ],
            "Resource": [
                "arn:aws:logs:*:*:log-group:/aws/amazonmq/*"
            ]
        }
    ]
}
```

您必須設定許可，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

您不需要手動建立服務連結角色，當您首次建立代理程式時，Amazon MQ 會建立服務連結角色來代表您呼叫 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 或 AWS API 中，使用服務名稱建立`mq.amazonaws.com`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。

> **Important**
>
> 服務連結角色僅針對 Amazon MQ for RabbitMQ 建立。

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 管理主控台、Amazon MQ CLI 或 Amazon MQ API 刪除 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
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

區域名稱	區域身分	Amazon MQ 支援
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
- 我想要允許 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

如果 AWS 管理主控台 告知您無權執行 動作，則必須聯絡您的管理員尋求協助。您的管理員是為您提供簽署憑證的人員。

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 可讓您將現有角色傳遞給該服務，而不是建立新的服務角色或服務連結角色。如需執行此作業，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 沒有將角色傳遞給服務的許可。

```
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。

如果您需要協助，請聯絡您的 AWS 管理員。您的管理員提供您的簽署憑證。

我想要允許 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物件取得該角色。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的資源的許可。

如需進一步了解，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 AWS 帳戶 在您擁有的 資源之間提供存取權，請參閱《IAM 使用者指南》中的在您擁有 AWS 帳戶 的另一個 IAM 使用者中提供存取權。
- 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 AWS 帳戶 擁有。
- 如需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)。
- 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 中的跨帳戶資源存取。

Amazon MQ 的合規驗證

在多個合規計畫中，第三方稽核人員會評估 Amazon MQ 的安全與 AWS 合規。這些包括 SOC、PCI、HIPAA 等。

若要了解 AWS 服務 是否在特定合規計劃範圍內，請參閱AWS 服務 合規計劃範圍內然後選擇您感興趣的合規計劃。如需一般資訊，請參閱AWS 合規計劃。

您可以使用 下載第三方稽核報告 AWS Artifact。如需詳細資訊，請參閱下載報告 in AWS Artifact

您使用 時的合規責任 AWS 服務 取決於資料的機密性、您公司的合規目標，以及適用的法律和法規。如需使用 時合規責任的詳細資訊 AWS 服務，請參閱 AWS 安全文件。

Amazon MQ 的恢復能力

AWS 全球基礎設施是以 AWS 區域和可用區域為基礎建置。 AWS 區域提供多個實體分隔和隔離的可用區域，這些可用區域以低延遲、高輸送量和高備援聯網連接。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、容錯能力和擴展能力，均較單一或多個資料中心的傳統基礎設施還高。

如需 AWS 區域和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

Amazon MQ 是受管服務，受到 AWS 全球網路安全的保護。如需 AWS 安全服務以及 如何 AWS 保護基礎設施的相關資訊，請參閱AWS 雲端安全。若要使用基礎設施安全的最佳實務設計您的 AWS 環境，請參閱安全支柱 AWS Well-Architected Framework 中的基礎設施保護。

您可以使用 AWS 已發佈的 API 呼叫，透過網路存取 Amazon MQ。使用者端必須支援下列專案：

- 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。

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 AWS 安全部落格上的如何透過減少攻擊面來協助準備 DDoS 攻擊。

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

為了提高私有代理程式的安全性，您應該透過正確設定 Amazon VPC 安全群組來限制不必要的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：

- 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 AWS 解決方案可靠性、可用性和效能的重要部分。您應該從 AWS 解決方案的所有部分收集監控資料，以便在發生多點故障時更輕鬆地偵錯。 AWS 提供數種工具來監控 Amazon MQ 資源並回應潛在事件：

您可以使用 CloudWatch 來檢視和分析 Amazon MQ 代理程式的指標。您可以從 CloudWatch 主控台、 AWS CLI或 CloudWatch AWS CLI檢視和分析代理程式指標。Amazon MQ 的 CloudWatch 指標會每分鐘自動從代理程式輪詢並推送到 CloudWatch。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 只會監控依消費者數量排序的前 500 個目的地。

如需 Amazon MQ 指標的完整清單，請參閱 適用於 ActiveMQ 代理程式的 Amazon MQ 可用 CloudWatch 指標。

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。

存取 Amazon MQ 的 CloudWatch 指標

您可以使用 AWS 管理主控台 AWS CLI、 和 API 存取 CloudWatch 指標。

您可能想要在沒有使用 的情況下存取 CloudWatch 指標 AWS 管理主控台。

若要使用 存取 Amazon MQ 指標 AWS CLI，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。

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 取得 CloudWatch 指標 AWS 管理主控台

下列範例說明如何使用 存取 Amazon MQ 的 CloudWatch 指標 AWS 管理主控台。如果您已登入 Amazon MQ 主控台，請在代理程式詳細資訊頁面上，選擇動作、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)。

存取 Prometheus 指標

> **Note**
>
> Prometheus 指標僅適用於 RabbitMQ 4.2 和更新版本。ActiveMQ 代理程式不支援 Prometheus 指標。

Amazon MQ 現在支援 Amazon MQ for RabbitMQ 代理程式的 Prometheus 指標。Prometheus 指標可讓您將中介裝置可觀測性整合至現有的監控基礎設施，讓您統一檢視中介裝置效能與其他 服務。使用 Prometheus 指標，您可以設定精細警示和儀表板，以主動偵測和回應簡訊工作負載中的問題。

從 RabbitMQ 4.2 開始，Amazon MQ for RabbitMQ 支援 Prometheus 指標，可讓您使用 Prometheus 監控系統來抓取代理程式指標。支援下列端點：

- `/metrics`
- `/metrics/detailed`
- `/metrics/memory-breakdown`

不支援`/metrics/per-object`端點。

如需每個端點公開指標的詳細資訊，請參閱 RabbitMQ 文件中的 Prometheus 指標。

Prometheus 指標與 CloudWatch 指標

Amazon MQ for RabbitMQ 透過 Prometheus 端點和 CloudWatch 公開指標。雖然兩者都提供代理程式運作狀態的可見性，但範圍和用量不同。

Prometheus 端點公開了一組更豐富的 RabbitMQ 代理程式運作狀態彙總指標，涵蓋更廣泛的代理程式內部，例如連線流失、頻道活動、佇列和交換統計資料，以及 Raft 共識指標。這些適用於與現有的 Prometheus 型監控基礎設施和精細警示整合。

CloudWatch 指標是從 Prometheus 端點取得的精選代理程式指標子集。如需可用 CloudWatch 指標的完整清單，請參閱 Amazon MQ for RabbitMQ 代理程式可用的 CloudWatch 指標。

在 CloudWatch 中，指標一律會在視覺化前以至少 60 秒的間隔進行彙總。相反地，Prometheus 公開原始指標資料點，而 Grafana 之類的儀表板解決方案可視覺化個別資料點，預設不會彙總。因此，相同指標的視覺化可能會根據 CloudWatch 中使用的統計資料，在 CloudWatch 和 Prometheus 之間產生差異

> **Note**
>
> 建議使用 Prometheus 對 Amazon MQ for RabbitMQ 操作指標進行非彙總監控。

取得和存取 Prometheus 端點

您可以使用 AWS 管理主控台 或 取得 Amazon MQ for RabbitMQ 代理程式的 Prometheus 端點 AWS CLI。

- AWS 管理主控台 — 導覽至 Amazon MQ 主控台，開啟代理程式的詳細資訊頁面，然後在連線區段下找到 Prometheus 端點。
- AWS CLI — 使用 `describe-broker`命令：

```
aws mq describe-broker --broker-id <broker-id>
```

  Prometheus 端點會在 下的回應中傳回`BrokerInstances.Endpoints`。

Amazon MQ for RabbitMQ Prometheus 支援使用與代理程式相同的身分驗證機制。如需支援的身分驗證方法的詳細資訊，請參閱 Amazon MQ for RabbitMQ 身分驗證和授權。若要了解如何在 Prometheus 中設定身分驗證，請參閱 Prometheus 文件中的 http_config。

Prometheus 組態最佳實務

- 設定 60 秒或更長的抓取期間。為了操作安全，建議這麼做。

抓取組態範例

下列各節提供 Amazon MQ for RabbitMQ 的範例 Prometheus 抓取組態。<broker-prometheus-endpoint> 將 取代為代理程式的 Prometheus 端點主機名稱，將 <username> 取代<password>為代理程式登入資料。

建議組態

對於大多數使用案例，建議使用下列組態。擴展/metrics端點可提供有關整體叢集運作狀態的良好彙總指標，讓您清楚檢視代理程式效能，而無須額外負荷詳細的指標集合。

```
global:
  scrape_interval: 60s

scrape_configs:
  - job_name: 'rabbitmq-aws-cluster'
    scheme: https
    basic_auth:
      username: <username>
      password: <password>
    metrics_path: '/metrics'
    static_configs:
      - targets:
        - '<broker-prometheus-endpoint>:16001'
        - '<broker-prometheus-endpoint>:16002'
        - '<broker-prometheus-endpoint>:16003'
```

詳細指標組態

下列組態會抓取其他詳細指標系列，以更深入地觀察特定代理程式元件。

```
global:
  scrape_interval: 60s

scrape_configs:
  - job_name: 'rabbitmq-connection-churn'
    scheme: https
```

```
    basic_auth:
      username: <username>
      password: <password>
    metrics_path: '/metrics/detailed'
    params:
      family: ['connection_churn_metrics']
    static_configs:
      - targets:
        - '<broker-prometheus-endpoint>:16001'
        - '<broker-prometheus-endpoint>:16002'
        - '<broker-prometheus-endpoint>:16003'
  - job_name: 'rabbitmq-ra'
    scheme: https
    basic_auth:
      username: <username>
      password: <password>
    metrics_path: '/metrics/detailed'
    params:
      family: ['ra_metrics']
    static_configs:
      - targets:
        - '<broker-prometheus-endpoint>:16001'
        - '<broker-prometheus-endpoint>:16002'
        - '<broker-prometheus-endpoint>:16003'
  - job_name: 'rabbitmq-queue'
    scheme: https
    basic_auth:
      username: <username>
      password: <password>
    metrics_path: '/metrics/detailed'
    params:
      family: ['queue_metrics']
    static_configs:
      - targets:
        - '<broker-prometheus-endpoint>:16001'
        - '<broker-prometheus-endpoint>:16002'
        - '<broker-prometheus-endpoint>:16003'
  - job_name: 'rabbitmq-exchange'
    scheme: https
    basic_auth:
      username: <username>
      password: <password>
    metrics_path: '/metrics/detailed'
    params:
```

```
      family: ['exchange_metrics']
    static_configs:
      - targets:
        - '<broker-prometheus-endpoint>:16001'
        - '<broker-prometheus-endpoint>:16002'
        - '<broker-prometheus-endpoint>:16003'
  - job_name: 'rabbitmq-connection'
    scheme: https
    basic_auth:
      username: <username>
      password: <password>
    metrics_path: '/metrics/detailed'
    params:
      family: ['connection_metrics']
    static_configs:
      - targets:
        - '<broker-prometheus-endpoint>:16001'
        - '<broker-prometheus-endpoint>:16002'
        - '<broker-prometheus-endpoint>:16003'
  - job_name: 'rabbitmq-channel'
    scheme: https
    basic_auth:
      username: <username>
      password: <password>
    metrics_path: '/metrics/detailed'
    params:
      family: ['channel_metrics']
    static_configs:
      - targets:
        - '<broker-prometheus-endpoint>:16001'
        - '<broker-prometheus-endpoint>:16002'
        - '<broker-prometheus-endpoint>:16003'
  - job_name: 'rabbitmq-exchange-count'
    scheme: https
    basic_auth:
      username: <username>
      password: <password>
    metrics_path: '/metrics/detailed'
    params:
      family: ['exchange_names']
    static_configs:
      - targets:
        - '<broker-prometheus-endpoint>:16001'
        - '<broker-prometheus-endpoint>:16002'
```

```
        - '<broker-prometheus-endpoint>:16003'
```

適用於 ActiveMQ 代理程式的 Amazon MQ 可用 CloudWatch 指標

Amazon MQ for ActiveMQ 指標

指標	單位	說明
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** 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。 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

指標	單位	說明
		CPU 使用率的大幅提升支付費用。 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。

指標	單位	說明
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。
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。

指標	單位	說明
TotalProducerCount	計數	生產者在目前代理程式目的地啟用的訊息數量。
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	說明
Broker	代理程式的名稱 **Note** 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	說明
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。 **Note** 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。

指標	單位	說明
ExpiredCount	計數	每分鐘因訊息過期而無法傳送的訊息數量。
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。 **Important** 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	說明
`Broker`	代理程式的名稱。 **Note** 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。

Amazon MQ for RabbitMQ 代理程式可用的 CloudWatch 指標

Warning

從 RabbitMQ 4.2 開始， `RabbitMQIOReadAverageTime`和 `RabbitMQIOWriteAverageTime`已棄用，不會發佈有意義的值。這些指標將在下一個主要 RabbitMQ 版本中從 CloudWatch 中移除。

RabbitMQ 代理程式指標

Note

開放原始碼 RabbitMQ 不建議將 管理外掛程式用於生產或長期監控。建議使用 Prometheus 從 RabbitMQ 4.2 之後查詢每個節點的指標。

指標	單位	說明
`ExchangeCount`	計數	代理程式上設定的交換總數。

指標	單位	說明
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。 **Important** 頻道的概念專屬於 AMQP 0-9-1。
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。 **Note** 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。

指標	單位	說明
PublishRate	計數	訊息發佈至代理程式的速率。 產生的數字代表取樣時每秒的訊息數目。 **Important** 此指標僅反映 AMQP 0-9-1 通訊協定活動。如需 AMQP 1.0 指標，請參閱 存取 Prometheus 指標。
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。 產生的數字代表取樣時每秒的訊息數目。 **Important** 此指標僅反映 AMQP 0-9-1 通訊協定活動。如需 AMQP 1.0 指標，請參閱 存取 Prometheus 指標。

指標	單位	說明
AckRate	計數	消費者認可訊息的速率。 產生的數字代表取樣時每秒的訊息數目。 **Important** 此指標僅反映 AMQP 0-9-1 通訊協定活動。如需 AMQP 1.0 指標，請參閱 存取 Prometheus 指標。
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。
RabbitMQMemLimit	位元組	RabbitMQ 代理程式的 RAM 限制。此指標會根據執行個體類型而有所不同。如需詳細資訊，請參閱the section called “記憶體和磁碟警示”。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。

指標	單位	說明
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。此指標會根據執行個體類型和部署模式而有所不同。如需詳細資訊，請參閱the section called “記憶體和磁碟警示”。
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。
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	說明
Broker	代理程式的名稱。

RabbitMQ 節點指標

指標	單位	說明
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。此指標會根據執行個體類型而有所不同。如需詳細資訊，請參閱the section called “記憶體和磁碟警示”。
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。此指標會根據執行個體類型和部署模式而有所不同。如需詳細資訊，請參閱the section called “記憶體和磁碟警示”。
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。

指標	單位	說明
ExchangeCount	計數	在節點上設定的交換總數。適用於 RabbitMQ 4.2 及更高版本。
QueueCount	計數	在節點上設定的佇列總數。適用於 RabbitMQ 4.2 及更高版本。
ConnectionCount	計數	在節點上建立的連線總數。適用於 RabbitMQ 4.2 及更高版本。
ChannelCount	計數	在節點上建立的頻道總數。適用於 RabbitMQ 4.2 及更高版本。
ConsumerCount	計數	連線至節點的消費者總數。適用於 RabbitMQ 4.2 及更高版本。
MessageCount	計數	節點上佇列中的訊息總數。適用於 RabbitMQ 4.2 及更高版本。
MessageReadyCount	計數	節點上佇列中的就緒訊息總數。適用於 RabbitMQ 4.2 及更高版本。
MessageUnacknowledgedCount	計數	節點上佇列中未確認的訊息總數。適用於 RabbitMQ 4.2 及更高版本。

從 RabbitMQ 節點指標彙總整個叢集的指標

若要取得彙總的整個叢集指標，您可以透過篩選代理程式名稱和指標名稱，在 CloudWatch 主控台上尋找對應的每個節點指標。然後，按一下核取方塊來選取這些指標，然後選擇新增數學 > 一般 > 總和。

RabbitMQ 節點指標的維度

維度	說明
Node	節點的名稱。 **Note** 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	說明
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。
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	說明
`Queue`	佇列的名稱。
`VirtualHost`	虛擬主機的名稱。
`Broker`	代理程式的名稱。

RabbitMQ 網路指標

指標	單位	說明
`NetworkOut`	位元組	執行個體在所有網路介面上送出的位元組數目。此指標識別來自單一執行個體之傳出網路流量的磁碟區。所報告的數目是在期間內送出的位元組總數。如果您要使用基本 (5 分鐘) 監控並且統計數字為總和，則可以將此數字除以 300，以找出每秒的位元組數。如果您具有詳細 (1 分鐘) 監控並且統計數字為總和，請將它除以 60。您也可以使用 CloudWatch 指標數學函數 `DIFF_TIME` ，尋找每秒位元組數。例如，如果您在 CloudWatch 中將 NetworkOut 繪製為 `m1`，指標數學公式會以位元組/秒為單位`m1/(DIFF_TIME(m1))` 傳回指標。如需 `DIFF_TIME` 和其他指標數學函數的詳細資訊，請參閱使用指標數學。 有意義的統計資料：總和、平均值、最小值、最大值
`NetworkIn`	位元組	執行個體在所有網路介面上收到的位元組數目。此指標識別流向單一執行個體之傳入網路流量的磁碟區。所報告的數目

指標	單位	說明
		是在期間內收到的位元組總數。如果您要使用基本 (5 分鐘) 監控並且統計數字為總和，則可以將此數字除以 300，以找出每秒的位元組數。如果您具有詳細 (1 分鐘) 監控並且統計數字為總和，請將它除以 60。您也可以使用 CloudWatch 指標數學函數 DIFF_TIME ，尋找每秒位元組數。例如，如果您在 CloudWatch 中將 NetworkIn 繪製為 m1，指標數學公式會以位元組/秒為單位m1/(DIFF_TIME(m1)) 傳回指標。如需 DIFF_TIME 和其他指標數學函數的詳細資訊，請參閱使用指標數學。 有意義的統計資料：總和、平均值、最小值、最大值

RabbitMQ 代理程式的維度

維度	說明
Broker	代理程式的名稱。

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”。

使用 記錄 Amazon MQ API 呼叫 AWS CloudTrail

Amazon MQ 已與 整合 AWS CloudTrail，此服務可提供使用者、角色或服務進行的 Amazon MQ 呼叫記錄 AWS 。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 的相關資訊，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 會啟用。發生支援的 Amazon MQ 事件活動時，它會記錄在 CloudTrail 事件中，並在事件歷史記錄中記錄其他服務 AWS 事件。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。您可以建立線索，以在 AWS 帳戶中持續記錄事件。根據預設，當您使用 建立線索時 AWS 管理主控台，線索會套用至所有 AWS 區域。追蹤會記錄所有 AWS 區域的事件，並將日誌檔案交付至指定的 Amazon S3 儲存貯體。您也可以設定其他 AWS 服務，以進一步分析和處理 CloudTrail 日誌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？
- 該請求是否由其他服務提出 AWS ？

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專案。

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 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。

> **Note**
>
> 當您開啟日誌並從 ActiveMQ Web 主控台發佈訊息時，訊息的內容會傳送至 CloudWatch 並顯示在日誌中。

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
- 將 CreateLogGroup 許可新增至 Amazon MQ 使用者
- 為 Amazon MQ 設定資源型政策。
- 預防跨服務混淆代理人

了解 CloudWatch Logs 中的記錄結構

您可以在建立代理程式或編輯代理程式時設定進階代理程式設定時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。

JSON

```
{
                    "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。

下列以資源為基礎的政策授予 `logs:CreateLogStream`和 `logs:PutLogEvents`的許可 AWS。

JSON

```
{
                        "Version":"2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Allow",
                                "Principal": { "Service":
 "mq.amazonaws.com" },
                                "Action": [ "logs:CreateLogStream",
 "logs:PutLogEvents" ],
                                "Resource": "arn:aws:logs:*:*:log-group:/aws/
amazonmq/*"
                            }
```

```
                ]
            }
```

此資源型政策必須使用 進行設定 AWS CLI ，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**
>
> 由於此範例使用 /aws/amazonmq/字首，因此每個區域每個 AWS 帳戶只需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問題，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 中 AWS，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。為了防止這種情況， AWS 提供工具，協助您保護所有 服務的資料，而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金鑰，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。

JSON

```
{
                        "Version":"2012-10-17",
                        "Statement": [
                            {
                            "Effect": "Allow",
                            "Principal": {
                                "Service": "mq.amazonaws.com"
                            },
                            "Action": [
                                "logs:CreateLogStream",
                                "logs:PutLogEvents"
                            ],
                            "Resource": "arn:aws:logs:*:*:log-group:/aws/
amazonmq/*",
                            "Condition": {
                                "StringEquals": {
                                "aws:SourceAccount": "123456789012",
                                "aws:SourceArn": "arn:aws:mq:us-
west-1:123456789012:broker:my-broker:123456789012"
                                }
                            }
                            }
                        ]
                        }
```

您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。

JSON

```
{
                            "Version":"2012-10-17",
                            "Statement": [
                            {
                                "Effect": "Allow",
                                "Principal": {
                                "Service": [
                                    "mq.amazonaws.com"
                                ]
                                },
```

```
                                "Action": [
                                "logs:CreateLogStream",
                                "logs:PutLogEvents"
                                ],
                                "Resource": "arn:aws:logs:*:*:log-group:/aws/
amazonmq/*",
                                "Condition": {
                                "ArnLike": {
                                    "aws:SourceArn":
 "arn:aws:mq:*:123456789012:broker:*"
                                },
                                "StringEquals": {
                                    "aws:SourceAccount": "123456789012"
                                }
                                }
                            }
                            ]
                        }
```

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。

使用 Amazon MQ 對 CloudWatch Logs 組態進行故障診斷

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。

Amazon MQ 的配額

本主題列出 Amazon MQ 內的限制。您可以針對特定 AWS 帳戶變更下列許多限制。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	Description
代理程式名稱	• 在您的帳戶中必須是唯一的 AWS 。 • 長度必須介於 1 至 50 個字元之間。 • 必須僅包含 ASCII 可列印字元集中指定的字元。 • 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	200
較小型代理程式每個通訊協定的線路層級連線	**Important** 不適用於 RabbitMQ 代理程式。

限制	Description
	`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** 不適用於 RabbitMQ 代理程式。 `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。

限制	Description
組態名稱	• 長度必須介於 1 至 150 個字元之間。 • 必須僅包含 ASCII 可列印字元集中指定的字元。 • 只能包含英數字元、破折號、句點、底線和波狀符號 (- . _ ~)。

限制	Description
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。

限制	Description
使用者名稱	• 長度必須介於 1 至 100 個字元之間。 • 必須僅包含 ASCII 可列印字元集中指定的字元。 • 只能包含英數字元、破折號、句點、底線和波狀符號 (- . _ ~)。 • 不得包含逗號 (,)。
密碼	• 長度必須介於 12 至 250 個字元之間。 • 必須僅包含 ASCII 可列印字元集中指定的字元。 • 必須包含至少 4 個唯一字元。 • 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	Description
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.m5.*` 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** 不適用於 RabbitMQ 代理程式。 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。	**Important** 不適用於 RabbitMQ 代理程式。 `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** 不適用於 RabbitMQ 代理程式。 `mq.m5.*` 執行個體類型代理程式為 50 GB。

API 調節

下列限流配額會針對每個 AWS 帳戶彙總所有 Amazon MQ APIs，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。如需一般故障診斷，請參閱 the section called “故障診斷：一般 Amazon MQ”。如需對特定引擎版本進行故障診斷，請參閱下列各節。

對 Amazon MQ 上的 ActiveMQ 進行故障診斷 Amazon MQ

故障診斷主題	說明
一般性問題的故障診斷	使用本節中的資訊來協助您診斷和解決在 Amazon MQ 代理程式上使用 ActiveMQ 時可能遇到的常見問題。
布魯克特_ENI_DELETED	當您刪除代理程式的彈性網路界面 (ENI) 時，Amazon MQ 上的 ActiveMQ 會發出BROKER_ENI_DELETED 警示。 Amazon MQ
BROKER_OOM	當代理程式因為記憶體容量不足而經歷重新啟動迴圈時，Amazon MQ 上的 ActiveMQ 將引發 BROKER_OOM 警示 Amazon MQ

針對 Amazon MQ 上的 RabbitMQ 進行故障診斷 Amazon MQ

故障診斷主題	說明
一般性問題的故障診斷	診斷使用 RabbitMQ 代理程式時可能遇到的常見問題。
RABBITMQ_MEMORY_ALARM	當 CloudWatch 指標 識別的代理程式記憶體使用量超過 識別的記憶體限制RabbitMQMemUsed 時，RabbitMQ 會發

故障診斷主題	說明
	出高記憶體警示RabbitMQMemLimit 。
RABBITMQ_INVALID_KMS_KEY	使用客戶受管 AWS KMS key(CMK) 建立的代理程式偵測到已停用 AWS Key Management Service (KMS) 金鑰時，Amazon MQ 上的 RabbitMQ 將引發 INVALID_KMS_KEY 關鍵動作必要程式碼。 Amazon MQ
ANIMALSMQ_INVALID_ASSUMEROLE	當 中指定的 IAM 角色 ARN 無法由 Amazon MQ 擔任時，Amazon MQ 上的 RabbitMQ 將引發 INVALID_ASSUMEROLE 關鍵動作必要程式碼。 aws.arns.assume_role_arn Amazon MQ
偵錯工具_INVALID_ARN_LDAP	當 LDAP 服務帳戶密碼 ARN 無效或無法存取時，Amazon MQ 上的 RabbitMQ 將引發 INVALID_ARN_LDAP 關鍵動作所需的程式碼。 Amazon MQ
偵錯工具_INVALID_ARN_HTTP	當 HTTP auth_backend 的 SSL 憑證或金鑰檔案ARNs 無效或無法存取時，Amazon MQ 上的 RabbitMQ 將引發 INVALID_ARN_HTTP 關鍵動作必要程式碼。 Amazon MQ

故障診斷主題	說明
RABBITMQ_INVALID_ARN_SSL	當 EXTERNAL auth_mechanism 的一或多個 CA 憑證信任存放區的 ARNs 無效或無法存取時，Amazon MQ 上的 RabbitMQ 將引發 INVALID_ARN_SSL 關鍵動作必要程式碼。 Amazon MQ
RABBITMQ_INVALID_ARN	當代理程式組態中的一或多個 ARNs 無效或無法存取時，Amazon MQ 上的 RabbitMQ 將引發 INVALID_ARN 關鍵動作必要程式碼。 Amazon MQ
RABBITMQ_DISK_ALARM	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。
RANDOMMQ_BROKER_NOT_UPGRADEABLE_TO_V4	在啟用傳統佇列或 Khepri 的代理程式上嘗試升級至 RabbitMQ 4 時，Amazon MQ 上的 RabbitMQ 將引發 RABBITMQ_BROKER_NOT_UPGRADEABLE_TO_V4 關鍵動作必要程式碼。

故障診斷：一般 Amazon MQ

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 網路的詳細資訊，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分組，因為 Amazon MQ 上的 ActiveMQ 和 Amazon MQ 上的 RabbitMQ 會使用不同的連接埠進行連線。 Amazon MQ

Amazon MQ 上的 ActiveMQ Amazon MQ

- 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 上的 RabbitMQ Amazon MQ

- 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。

ActiveMQ on Amazon MQ

在 Amazon ActiveMQ上測試 ActiveMQ Amazon MQ

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>
        ...
```

RabbitMQ on Amazon MQ

在 Amazon MQ 代理程式的網路連線上測試 RabbitMQ Amazon MQ

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有存取性的 Amazon MQ 代理程式上的 RabbitMQ。Amazon MQ

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 受管 IAM 政策，AmazonMQFullAccess或在自訂 IAM 政策中擁有一組正確的 Amazon EC2 許可。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 “排程代理程式維護”。
- 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.t3.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 上的 ActiveMQ Amazon MQ – 檢查代理程式目前使用的已配置 Amazon EC2 運算單位`CpuUtilization`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 上的 RabbitMQ Amazon MQ – 檢查代理程式目前使用的已配置 Amazon EC2 運算單位`SystemCpuUtilization`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為您的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。

對 Amazon MQ 上的 ActiveMQ 進行故障診斷 Amazon MQ

使用本節中的資訊來協助您診斷和解決在 Amazon MQ 代理程式上使用 ActiveMQ 時可能遇到的常見問題。

內容

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。

如需在 Amazon MQ 上設定 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：

- 在 CloudWatch Logs 中檢查 Amazon MQ 代理程式上 ActiveMQ 的一般代理程式日誌。 Amazon MQ 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 代理程式上 ActiveMQ 的 CloudWatch Logs 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 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 上的 RabbitMQ Amazon MQ

使用本節中的資訊來協助您診斷和解決在 Amazon MQ 代理程式上使用 RabbitMQ 時可能遇到的常見問題。

內容

- 我無法變更代理程式的 Amazon VPC 組態。
- 叢集部署已暫停我的佇列同步。
- 我的 Amazon MQ for RabbitMQ 單一執行個體代理程式正在重新啟動迴圈中。
- 我失去了代理程式上所有管理員帳戶的存取權。

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，請檢查您的佇列或虛擬主機名稱是否包含任何空格、標籤或其他非 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 上的 RabbitMQ 中啟用外掛程式？

Amazon MQ 上的 RabbitMQ 目前僅支援依預設啟用的 RabbitMQ 管理、shovel、聯合、一致性雜湊交換外掛程式。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。

叢集部署已暫停我的佇列同步。

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”。

我的 Amazon MQ for RabbitMQ 單一執行個體代理程式正在重新啟動迴圈中。

引發高記憶體警示的 Amazon MQ for RabbitMQ 單一執行個體代理程式，如果重新啟動且沒有足夠的記憶體可供啟動，則會有無法使用的風險。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，您將無法套用 Amazon MQ 建議的最佳實務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，您可以在遇到高記憶體警示時升級單一執行個體代理程式，因為重新啟動期間節點之間沒有要執行的佇列同步。

我失去了代理程式上所有管理員帳戶的存取權。

您可以使用 IAM 身分驗證來復原存取權。啟用 AWS 帳戶的傳出 Web 聯合身分、建立具有取得 Web 身分字符許可的 IAM 角色、將代理程式設定為透過 OAuth 2.0 接受 IAM 身分驗證，然後使用 IAM 憑證取得 JWT 字符並建立新的管理員使用者。如需詳細說明，請參閱 the section called “使用 IAM 身分驗證和授權”。

Amazon MQ 上的 ActiveMQ：已刪除彈性網路介面警示 Amazon MQ

當您刪除代理程式的彈性網路界面 (ENI) 時，Amazon MQ 上的 ActiveMQ 將引發 BROKER_ENI_DELETED 警示。 Amazon MQ 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 上的 ActiveMQ：中介裝置記憶體不足警示 Amazon MQ

當代理程式因為記憶體容量不足而經歷重新啟動迴圈時，Amazon MQ 上的 ActiveMQ 會引發 BROKER_OOM 警示。 Amazon MQ 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	

Amazon MQ CloudWatch 指標	記憶體使用較高的原因	
	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。	
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。您可以進一步了解如何在 Amazon MQ 上的 ActiveMQ 最佳實務中確保有效效能。

Amazon MQ for RabbitMQ：高記憶體警示

當 CloudWatch 指標 識別的代理程式記憶體使用量RabbitMQMemUsed超過 識別的記憶體限制時，Amazon MQ for RabbitMQ 將發出高記憶體警示RabbitMQMemLimit。

引發高記憶體警示的 RabbitMQ 代理程式會封鎖所有發佈訊息的用戶端。您的代理程式可能會進入重新啟動迴圈、遇到暫停的佇列同步，或發生使警示診斷和解決複雜化的其他問題。

若要診斷和解決高記憶體警示，請先遵循 RabbitMQ 的所有最佳實務，然後完成下列步驟。

> **Important**
>
> - `RabbitMQMemLimit` 由 Amazon MQ 設定，並特別根據每個主機執行個體類型的可用記憶體進行調整。如需詳細資訊，請參閱the section called “記憶體和磁碟警示”。
> - 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)。

步驟 1：診斷高記憶體警示

診斷 Amazon MQ for RabbitMQ 代理程式的高記憶體警示有兩種方式。建議您在 CloudWatch 中檢查 RabbitMQ Web 主控台和 Amazon MQ 指標。

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 管理主控台 並開啟代理程式的 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。如需可透過 RabbitMQ Web 主控台取得之記憶體用量詳細資訊的詳細資訊，請參閱 RabbitMQ 伺服器文件網站上的記憶體使用原因。

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	

Amazon MQ CloudWatch 指標	記憶體使用較高的原因	
	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。	

步驟 2：解決和防止高記憶體警示

> **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。

遵循 RabbitMQ 的所有最佳實務作為一般預防方法。對於您識別的每個特定貢獻者，我們建議採取以下一組動作來解決和防止 RabbitMQ 高記憶體警示。

高記憶體使用的來源	用於定址的 Amazon MQ 建議	預防 的 Amazon MQ 建議
訊息數量	使用發佈至佇列的訊息、從佇列清除訊息，或從代理程式刪除佇列。	啟用延遲佇列，並設定或減少佇列深度限制。
佇列數量	減少佇列數量。	設定或減少佇列計數限制。
連線數量	減少連線數量。	設定或減少連線計數限制。
頻道數量	減少頻道數量。	在用戶端應用程式上設定每個連線的最大通道數量。
取用者數量	減少連線至代理程式的取用者數量。	設定小型取用者預擷取限制。
訊息發佈率	降低發佈者傳送訊息至代理程式的速率。	開啟發佈者確認。
用戶端連線嘗試率	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。	使用壽命較長的連線可減少連線嘗試次數和頻率。

代理程式的記憶體警示解決後，您可以將主機執行個體類型升級至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資訊，請參閱《Amazon MQ REST API 參考`UpdateBrokerInput`》中的 。

> **Note**
>
> 您無法將代理程式從`mq.m5.x`執行個體類型降級為`mq.t3.micro`執行個體類型。若要降級，您必須刪除代理程式並建立新的代理程式。

Amazon MQ 上的 RabbitMQ：無效的 AWS Key Management Service 金鑰 Amazon MQ

使用客戶受管 AWS KMS key(CMK) 建立的代理程式偵測到已停用 AWS Key Management Service (KMS) 金鑰時，Amazon MQ 上的 RabbitMQ 將引發 INVALID_KMS_KEY 關鍵動作必要程式碼。Amazon MQ 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 命令列界面 (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 上的 RabbitMQ：磁碟限制警示 Amazon MQ

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. 如需詳細資訊，請參閱the section called “記憶體和磁碟警示”。

已引發磁碟限制警示的 Amazon MQ 代理程式上的 RabbitMQ 將無法發佈新訊息。 Amazon MQ 如果您在相同的連線上有發佈者和消費者，消費者也將無法接收訊息。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：

- 建立新的連線以使用發佈至佇列的訊息。
- 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`。我們也建議您讓發佈者和消費者保持不同的連線。

Amazon MQ for RabbitMQ：執行個體類型變更警示

`RABBITMQ_CLUSTER_DISK_USAGE_TOO_HIGH_FOR_INSTANCE_CHANGE` 表示請求的代理程式執行個體類型變更無法繼續，因為目前 RabbitMQ 節點上的磁碟使用率很高。Amazon MQ for RabbitMQ 會在目前的磁碟用量超過 CloudWatch 指標 所識別的請求執行個體類型上可用的磁碟用量時發出此警示`RabbitMQDiskFree`。

進入 `RABBITMQ_CLUSTER_DISK_USAGE_TOO_HIGH_FOR_INSTANCE_CHANGE` 狀態的 RabbitMQ 代理程式將繼續可供您的應用程式使用，但不會繼續請求的執行個體類型變更。Amazon MQ 允許代理程式在此狀態下重新啟動，但當磁碟用量保持高於所請求執行個體類型的閾值時，您無法變更執行個體類型。代理程式會傳回 `ModifyBroker` API 操作的例外狀況，在此狀態下嘗試變更執行個體類型。

診斷和解決執行個體類型變更警示

預設情況下，Amazon MQ 為您的代理程式啟用指標。您可以存取 CloudWatch 主控台或使用 CloudWatch API 來檢視代理程式指標。 `MessageCount`和 `RabbitMQDiskFree`指標可用來診斷 `RABBITMQ_CLUSTER_DISK_USAGE_TOO_HIGH_FOR_INSTANCE_CHANGE`。

若要解決隔離狀態並允許執行個體類型變更繼續，請使用 Amazon MQ 管理主控台：

- 建立新的連線以使用發佈至佇列的訊息。
- 從佇列清除訊息。
- 從您的代理程式中刪除佇列。

> **Note**
>
> 在您採取必要的動作
> 後，RABBITMQ_CLUSTER_DISK_USAGE_TOO_HIGH_FOR_INSTANCE_CHANGE狀態可能需要數小時才能清除。

Amazon MQ 上的 RabbitMQ：無效的 IAM 擔任角色 Amazon MQ

當 中指定的 IAM 角色 ARN `aws.arns.assume_role_arn` 無效或無法由 Amazon MQ 擔任時，Amazon MQ 上的 RabbitMQ 將引發 INVALID_ASSUMEROLE 關鍵動作必要程式碼。當角色不存在、與代理程式位於不同的 AWS 帳戶中，或缺乏與 mq.amazonaws.com 的必要信任關係時，就會發生這種情況。

RABBITMQ_INVALID_ASSUMEROLE 隔離中的代理程式無法擷取 LDAP 身分驗證所需的登入資料或憑證，導致 LDAP 身分驗證無法使用。如果 LDAP 是唯一設定的身分驗證方法，使用者將無法連線到代理程式。Amazon MQ 需要 IAM 角色才能存取代理程式組態中 ARNs 參考 AWS 的資源，例如用於 LDAP 身分驗證的 AWS Secrets Manager 秘密或 Amazon S3 物件。

診斷和解決 RABBITMQ_INVALID_ASSUMEROLE

若要診斷和解決 RABBITMQ_INVALID_ASSUMEROLE 動作所需的程式碼，您必須使用 Amazon CloudWatch Logs 和 AWS Identity and Access Management 主控台。

解決無效的擔任角色問題

1. 導覽至 Amazon CloudWatch Logs Insights，並針對代理程式的日誌群組 執行下列查詢`/aws/amazonmq/broker/<broker-id>/general`：

```
fields @timestamp, @message
| sort @timestamp desc
| filter @message like /error.*aws_arn_config/
| limit 10000
```

2. 尋找類似以下的錯誤訊息：

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

3. 檢查 IAM 角色組態並修正下列問題：
   - 確保角色與代理程式位於相同的 AWS 帳戶中
   - 驗證信任政策允許 mq.amazonaws.com 擔任該角色
   - 確認角色具有存取所需 AWS 資源的適當許可
4. 更新代理程式組態並重新啟動代理程式。

Amazon MQ 上的 RabbitMQ：無效的 LDAP ARN Amazon MQ

當為 LDAP 服務帳戶密碼設定的 ARN 無效或無法存取時，Amazon MQ 上的 RabbitMQ 將引發 INVALID_ARN_LDAP 關鍵動作必要程式碼。 Amazon MQ 這適用於 `aws.arns.auth_ldap.dn_lookup_bind.password`或 中指定的 ARNs`aws.arns.auth_ldap.other_bind.password`，其必須參考包含純文字密碼的 AWS Secrets Manager 秘密。

RABBITMQ_INVALID_ARN_LDAP 隔離中的代理程式無法向 LDAP 服務帳戶進行身分驗證，導致 LDAP 身分驗證無法使用。如果 LDAP 是唯一設定的身分驗證方法，使用者將無法連線到代理程式。無效的 ARNs可能是由格式不正確的 ARN 語法、對不存在秘密的參考、與代理程式位於不同 AWS 區域的秘密，或 IAM 角色中的 Secretsmanager：GetSecretValue 許可不足所造成。

診斷和解決 RABBITMQ_INVALID_ARN_LDAP

若要診斷和解決 RABBITMQ_INVALID_ARN_LDAP 動作所需的程式碼，您必須使用 Amazon CloudWatch Logs 和 主控台。

解決無效的 LDAP ARN 問題

1. 導覽至 Amazon CloudWatch Logs Insights，並針對代理程式的日誌群組 執行下列查詢`/aws/amazonmq/broker/<broker-id>/general`：

```
fields @timestamp, @message
| sort @timestamp desc
| filter @message like /error.*aws_arn_config/
| limit 10000
```

2. 尋找類似以下的錯誤訊息：

```
[error] <0.254.0> aws_arn_config: {<<"could not resolve
 ARN 'arn:aws:secretsmanager:xxx' for configuration
 'aws.arns.auth_ldap.dn_lookup_bind.password', error: \"AWS service is unavailable
\"">>,{error,"AWS service is unavailable"}}
```

3. 檢查 Secrets Manager 秘密並修正任何問題，例如：

   - 確認秘密與代理程式位於相同的 AWS 區域中
   - 確認 ARN 語法正確
   - 確保 IAM 角色具有 secretsmanager：GetSecretValue 許可

4. 更新代理程式組態並重新啟動代理程式。

Amazon MQ 上的 RabbitMQ：無效的 HTTP ARN Amazon MQ

當 HTTP auth_backend 的一個或多個 ARNs 無效或無法存取時，Amazon MQ 上的 RabbitMQ 將引發 INVALID_ARN_HTTP 關鍵動作必要程式碼。 Amazon MQ 這適用於 `aws.arns.auth_http.ssl_options.cacertfile`、`aws.arns.auth_http.ssl_options.certfile`或 中指定的 ARN`aws.arns.auth_http.ssl_options.keyfile`，其必須參考包含憑證和私有金鑰的 Amazon S3 物件和 AWS Secrets Manager 秘密。

RABBITMQ_INVALID_ARN_HTTP 隔離中的代理程式無法透過 HTTP 伺服器進行身分驗證。如果 HTTP 是唯一設定的身分驗證方法，使用者將無法連線到代理程式。無效的 ARNs可能是由格式不正確的 ARN 語法、對不存在秘密的參考、與代理程式位於不同 AWS 區域的秘密，或 IAM 角色中的 s3：GetObject/secretsmanager：GetSecretValue 許可不足所造成。

診斷和解決 RABBITMQ_INVALID_ARN_HTTP

若要診斷和解決 RABBITMQ_INVALID_ARN_HTTP 動作所需的程式碼，您必須使用 Amazon CloudWatch Logs 和 主控台。

解決無效的 HTTP ARN 問題

1. 導覽至 Amazon CloudWatch Logs Insights，並針對代理程式的日誌群組 執行下列查詢/aws/amazonmq/broker/<broker-id>/general：

```
fields @timestamp, @message
| sort @timestamp desc
| filter @message like /error.*aws_arn_config/
| limit 10000
```

2. 尋找類似以下的錯誤訊息：

```
[error] <0.209.0> aws_arn_config: {<<"could not resolve ARN 'arn:aws:s3:::xxxx' for
 configuration 'aws.arns.auth_http.ssl_options.certfile', error: \"AWS service is
 unavailable\"">>,{error,"AWS service is unavailable"}}
```

3. 檢查 S3 Object/Secrets Manager 秘密並修正任何問題，例如：
   - 確認資源與代理程式位於相同的 AWS 區域中
   - 確認 ARN 語法正確
   - 確保 IAM 角色具有 s3：GetObject 和 secretsmanager：GetSecretValue 許可
4. 更新代理程式組態並重新啟動代理程式。

Amazon MQ 上的 RabbitMQ：無效的 SSL ARN Amazon MQ

當 EXTERNAL auth_mechanism 的一或多個 CA 憑證信任存放區的 ARNs 無效或無法存取時，Amazon MQ 上的 RabbitMQ 將引發 INVALID_ARN_SSL 關鍵動作必要程式碼。 Amazon MQ 這適用於 `aws.arns.ssl_options.cacertfile`或 中指定的 ARN`Saws.arns.management.ssl.cacertfile`，其必須參考包含憑證的 Amazon S3 或 ACM PCA 物件。

RABBITMQ_INVALID_ARN_SSL 隔離中的代理程式無法在交互 TLS 交握期間驗證用戶端憑證，因為未設定有效的信任存放區。如果 EXTERNAL 驗證機制是唯一設定的身分驗證方法，使用者將無法連線到代理程式。無效的 ARNs可能是由格式不正確的 ARN 語法、對不存在 S3 物件的

參考、與代理程式位於不同 AWS 區域的 S3 物件，或 IAM 角色中的 s3：GetObject/acm-pca：GetCertificateAuthorityCertificate 許可不足所造成。

診斷和解決 RABBITMQ_INVALID_ARN_SSL

若要診斷和解決 RABBITMQ_INVALID_ARN_SSL 動作所需的程式碼，您必須使用 Amazon CloudWatch Logs 和 主控台。

解決無效的 SSL ARN 問題

1. 導覽至 Amazon CloudWatch Logs Insights，並針對代理程式的日誌群組 執行下列查詢`/aws/amazonmq/broker/<broker-id>/general`：

```
fields @timestamp, @message
| sort @timestamp desc
| filter @message like /error.*aws_arn_config/
| limit 10000
```

2. 尋找類似以下的錯誤訊息：

```
[error] <0.209.0> aws_arn_config: {<<"could not resolve ARN 'arn:aws:acm-pca:xxxx'
 for configuration 'aws.arns.ssl_options.cacertfile', error: \"AWS service is
 unavailable\"">>,{error,"AWS service is unavailable"}}
```

3. 檢查 S3/ACM-PCA 物件並修正任何問題，例如：
   - 確認秘密與代理程式位於相同的 AWS 區域中
   - 確認 ARN 語法正確
   - 確保 IAM 角色具有 s3：GetObject/acm-pca：GetCertificateAuthorityCertificate 許可
4. 更新代理程式組態並重新啟動代理程式。

Amazon MQ 上的 RabbitMQ：無效的 ARN Amazon MQ

當中介裝置中設定的一或多個 ARNs 無效或無法存取時，Amazon MQ 上的 RabbitMQ 將引發 INVALID_ARN 關鍵動作必要程式碼。 Amazon MQ 這適用於用於 SSL 憑證、 AWS Secrets Manager

秘密、Amazon S3 物件或其他 AWS 資源參考ARNs，這些資源參考未被更具體的隔離代碼涵蓋，例如 RABBITMQ_INVALID_ARN_LDAP 或 RABBITMQ_INVALID_ASSUMEROLE。

RABBITMQ_INVALID_ARN 隔離中的代理程式可能會遇到功能降低，具體取決於哪些 ARNs 無效。相依於無法存取資源的功能將無法使用，且代理程式會記錄錯誤，指出哪個 ARN 無法解析。對代理程式可用性的影響取決於關鍵代理程式操作是否需要無效的 ARN。

診斷和解決 RABBITMQ_INVALID_ARN

若要診斷和解決 RABBITMQ_INVALID_ARN 動作所需的程式碼，您必須為受影響的資源使用 Amazon CloudWatch Logs 和適當的 AWS 服務主控台。

解決無效的 ARN 問題

1. 導覽至 Amazon CloudWatch Logs Insights，並針對代理程式的日誌群組 執行下列查詢`/aws/amazonmq/broker/<broker-id>/general`：

```
fields @timestamp, @message
| sort @timestamp desc
| filter @message like /error.*aws_arn_config/
| limit 10000
```

2. 尋找類似以下的錯誤訊息：

```
[error] <0.254.0> aws_arn_config: {<<"could not resolve ARN
 'arn:aws:s3:::bucket-name/certificate.pem' for configuration
 'aws.arns.auth_ldap.ssl_options.cacertfile', error: \"AWS service is unavailable
\"">>,{error,"AWS service is unavailable"}}
```

3. 檢查 AWS 資源並修正任何問題，例如：
   - 確認資源與代理程式位於相同的 AWS 區域中
   - 確認 ARN 語法正確
   - 確保 IAM 角色具有適當的許可來存取資源
4. 更新代理程式組態並重新啟動代理程式。

Amazon MQ for RabbitMQ：中介裝置無法升級至第 4 版

當您嘗試將 RabbitMQ 3 代理程式升級至 RabbitMQ 4，且代理程式具有傳統佇列或啟用 Khepri 中繼資料存放區功能旗標時，Amazon MQ for RabbitMQ 將引發`RABBITMQ_BROKER_NOT_UPGRADEABLE_TO_V4`動作所需的程式碼。 RabbitMQ Amazon MQ 不會套用主要版本升級，並將讓代理程式可供發佈和使用。

此動作所需的程式碼僅適用於 RabbitMQ 3 代理程式。若要解決此狀態並繼續升級，請完成下列步驟。

診斷和解決 RABBITMQ_BROKER_NOT_UPGRADEABLE_TO_V4

1. 使用 Amazon MQ 佇列遷移工具將所有傳統佇列遷移至規定人數佇列。工具可透過 RabbitMQ Web 主控台 (管理員 > 佇列遷移) 或透過 HTTP API 存取。
2. 如果在代理程式上啟用 Khepri，就沒有 RabbitMQ 4 的就地升級路徑。請改為考慮 RabbitMQ 藍綠部署。

解決基礎問題後，Amazon MQ 會自動清除`CRITICAL_ACTION_REQUIRED`狀態。

> **Note**
>
> 您可以使用 UpdateBroker API 操作將代理程式引擎版本更新回 3.13，以清除`CRITICAL_ACTION_REQUIRED`狀態。

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	您可以用來處理訊息中介裝置的 AWS CLI 命令描述。
AWS CloudFormation 《 使用者指南》中的 Amazon MQ	`AWS::AmazonMQ::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。 `AWS::AmazonMQ::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	有關 AWS Premium Support 的資訊的主要網頁，這是one-on-one、快速回應的支援管道，可協助您在基礎設施服務上 AWS 建置和執行應用程式

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。

Date	文件更新
2026 年 5 月 5 日	Amazon MQ 現在支援從 RabbitMQ 3.13 到 RabbitMQ 4.2 的就地主要版本升級。您的中介裝置端點保持不變，因此不需要變更應用程式碼。在升級期間，Amazon MQ 會封鎖所有連線以執行安全升級，進而導致代理程式停機。 如需詳細資訊，請參閱 • 從 RabbitMQ 3.x 升級到 4.x
2026 年 4 月 30 日	Amazon MQ for RabbitMQ 現在支援執行 RabbitMQ 4.2 和更新版本的代理程式 Prometheus 指標，可讓您將代理程式可觀測性整合至現有的 Prometheus 型監控基礎設施。支援下列端點：`/metrics`、`/metrics/detailed` 和 `/metrics/memory-breakdown` 。 如需詳細資訊，請參閱存取 Prometheus 指標和 Amazon MQ for RabbitMQ 代理程式的可用 CloudWatch 指標。
2026 年 2 月 19 日	Amazon MQ Amazon MQ 現在支援 ActiveMQ 5.19，這是新的次要引擎版本版本。 如需詳細資訊，請參閱 • ActiveMQ 5.19 版本頁面 • 管理 Amazon MQ for ActiveMQ 引擎版本 • 升級 Amazon MQ 代理程式引擎版本 • 使用 Spring XML 組態檔案
2026 年 1 月 22 日	Amazon MQ 現在支援適用於 RabbitMQ 4.2 及更高版本的代理程式的 JMS 主題交換外掛程式。您可以使用官方 RabbitMQ JMS 用戶端在 Amazon MQ for RabbitMQ 代理程式上執行 JMS 工作負載。它支援 JMS 1.1、2.0 和 3.1。 如需詳細資訊，請參閱 • 官方 JMS 2.0 規格 （與 和擴充的 JMS 1.1 回溯相容）

Date	文件更新
	• 官方 JMS 3.1 規格 • RabbitMQ JMS 用戶端的限制 • 將您的 JMS 應用程式連線至 Amazon MQ for RabbitMQ 代理程式
2026 年 1 月 8 日	Amazon MQ 現在支援在 RabbitMQ 4.2 及更高版本上使用 X.509 用戶端憑證的代理程式 SSL 憑證驗證，以及交互 TLS (mTLS) 組態。您可以透過 AWS 管理主控台 AWS CloudFormation AWS CLI、或在可使用 Amazon MQ 的所有 AWS 區域 AWS CDK 中設定 SSL 憑證身分驗證和 mTLS。 如需詳細資訊，請參閱 SSL 憑證身分驗證和設定 mTLS。
2026 年 1 月 6 日	Amazon MQ 現在支援使用外部 HTTP 伺服器在 RabbitMQ 4.2 及更高版本上執行代理程式的 HTTP 身分驗證和授權。您可以在可使用 Amazon MQ 的所有 AWS CDK 中 AWS 管理主控台，透過 AWS CloudFormation AWS CLI或 AWS 區域 設定 HTTP 身分驗證。 如需詳細資訊，請參閱 HTTP 身分驗證和授權。
2025 年 11 月 20 日	Amazon MQ 現在支援 RabbitMQ 4.2，這是一個新的主要版本版本，引進了對 AMQP 1.0 通訊協定的原生支援、以 Raft 為基礎的新中繼資料存放區 Khepri、本機鏟子，以及規定人數佇列的訊息優先順序。RabbitMQ 4.2 也包含輸送量和記憶體管理的各種錯誤修正和效能改進。雖然此版本推出新功能，但有一些重大變更。 如需詳細資訊，請參閱 • RabbitMQ 4 • 開放原始碼 RabbitMQ 版本備註 • 設定資源限制 • 支援的通訊協定 • Amazon MQ 版本升級
2025 年 11 月 18 日	Amazon MQ 現在支援適用於 RabbitMQ 的 Graviton3 支援的 m7g 執行個體，其大小範圍從非洲 （開普敦） 的中型到 16xlarge。 如需詳細資訊，請參閱Amazon MQ for RabbitMQ 代理程式執行個體類型。

Date	文件更新
2025 年 11 月 17 日	Amazon MQ 現在支援使用外部 LDAP 目錄服務的 RabbitMQ 代理程式的 LDAP 身分驗證和授權。您可以在可使用 Amazon MQ 的所有 AWS CDK 中 AWS 管理主控台 AWS CloudFormation，透過 AWS CLI或 AWS 區域 設定 LDAP。 如需詳細資訊，請參閱Amazon MQ for RabbitMQ 的 LDAP 身分驗證和授權。
2025 年 10 月 22 日	Amazon MQ 現已在亞太區域 （紐西蘭） 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」。
2025 年 9 月 3 日	Amazon MQ 現在支援具有公有身分提供者 (IdPs) 的 RabbitMQ 代理程式的 OAuth 2.0 身分驗證和授權。您可以透過 AWS 管理主控台 AWS CloudFormation AWS CLI、或在可使用 Amazon MQ 的所有 AWS 區域 AWS CDK 中設定 OAuth 2.0。 如需詳細資訊，請參閱Amazon MQ for RabbitMQ 的 OAuth 2.0 身分驗證和授權。
2025 年 7 月 22 日	Amazon MQ 現在支援 Graviton3 支援的 RabbitMQ `m7g`執行個體，其大小範圍從中型到 16xlarge。在`m7g`執行個體上執行的 RabbitMQ 叢集與在`m5`執行個體上執行的同等 Amazon MQ for RabbitMQ 叢集相比，可提供高達 50% 的工作負載容量和高達 85% 的輸送量改善。 `M7g`執行個體也有最佳化磁碟區大小，因執行個體大小而異。如需詳細資訊，請參閱Broker instance types。 `M7g` Amazon MQ 上的 執行個體現在可在非洲 （開普敦）、加拿大西部 （卡加利） 和歐洲 （米蘭） 區域以外的所有一般可用區域中使用。
2025 年 7 月 8 日	Amazon MQ 現已在亞太區域 （台北） 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」。
2025 年 4 月 22 日	您現在可以使用 `DeleteConfiguration` API 刪除 Amazon MQ 代理程式組態。如需詳細資訊，請參閱《Amazon MQ API 參考》中的組態。

Date	文件更新
2025 年 4 月 16 日	Amazon MQ for RabbitMQ 現在支援使用雙堆疊 (IPv4 和 IPv6) 端點連線至公有和私有代理程式。如需詳細資訊，請參閱Connecting to Amazon MQ及Configuring a private Amazon MQ broker。
2025 年 4 月 7 日	Amazon MQ 現已在亞太區域 （泰國） 和墨西哥 （中部） 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」。
2025 年 2 月 13 日	Amazon MQ API FIPS 端點現在可在加拿大 （中部） 和加拿大西部 （卡加利） 區域使用。 如需搭配 Amazon MQ API 使用 FIPS 端點的詳細資訊，請參閱 Connecting to Amazon MQ。 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」。
2025 年 2 月 12 日	Amazon MQ 宣布下列執行個體類型終止支援日期： Broker instance types • ActiveMQ`mq.t2.micro` ：2025 年 5 月 12 日 • ActiveMQ`mq.m4.large` ：2025 年 5 月 12 日 您無法在 2025 年 3 月 17 日`mq.m4.large` 當天`mq.t2.micro` 或之後建立代理程式。
2024 年 12 月 10 日	Amazon MQ 現在支援使用 AWS PrivateLink 在虛擬私有雲端 (VPCs) 與 Amazon MQ API 之間連線，而無需將您的流量暴露到公有網際網路。如需詳細資訊，請參閱the section called “使用 AWS PrivateLink 連線至 Amazon MQ”。
2024 年 11 月 18 日	Amazon MQ 現已在亞太區域 （馬來西亞） 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」。

Date	文件更新
2024 年 11 月 14 日	Amazon MQ 宣布下列引擎版本終止支援日期： 管理 Amazon MQ for ActiveMQ 引擎版本 • ActiveMQ 5.17：2025 年 6 月 16 日 管理 Amazon MQ for RabbitMQ 引擎版本 • RabbitMQ 3.11：2025 年 2 月 17 日 • RabbitMQ 3.12：2025 年 3 月 17 日 如需升級至最新版本的詳細資訊，請參閱 升級 Amazon MQ 代理程式引擎版本
2024 年 11 月 13 日	Amazon MQ 現在支援雙堆疊服務端點，您可以使用 IPv4 或 IPv6 連線到這些端點。Amazon MQ 雙堆疊區域服務端點可以使用 A和 AAAA DNS 記錄來解析。如需詳細資訊，請參閱???。
2024 年 7 月 25 日	Amazon MQ 現在支援 ActiveMQ 5.18，這是新的次要引擎版本版本。如需詳細資訊，請參閱下列內容： • ActiveMQ 5.18 版本頁面 • 管理 Amazon MQ for ActiveMQ 引擎版本 • 升級 Amazon MQ 代理程式引擎版本 • 使用 Spring XML 組態檔案
2024 年 7 月 22 日	Amazon MQ 現在僅在使用 3.13 版及更高版本的代理程式上支援規定人數佇列。配額佇列是一種複寫的 FIFO 佇列類型，使用 Raft 共識演算法來維持資料一致性。配額佇列提供有毒訊息處理，可協助您管理未處理的訊息。 若要開始使用規定人數佇列，請參閱 Amazon MQ 上 RabbitMQ 的配額佇列 Amazon MQ。

Date	文件更新
2024 年 7 月 2 日	Amazon MQ for RabbitMQ 現在支援 RabbitMQ 3.13，這是一個次要版本版本。對於使用引擎 3.13 版及更高版本的所有代理程式，Amazon MQ 會在維護時段期間管理升級至最新支援的修補程式版本。如需詳細資訊，請參閱升級 Amazon MQ 代理程式引擎版本。 Amazon MQ for RabbitMQ 大小調整準則 已更新，納入新的佇列限制、每個頻道的消費者限制，以及使用引擎 3.13 版的代理程式鏟。 如需此版本中修正和功能的詳細資訊，請參閱 RabbitMQ 伺服器 GitHub 儲存庫上的 RabbitMQ 3.13 版本備註。 RabbitMQ GitHub 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 6 月 10 日	Amazon MQ 現已在加拿大西部（卡加利）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」。
2024 年 5 月 10 日	Amazon MQ 版本支援行事曆會指出代理程式引擎版本何時結束支援。當引擎版本終止支援時，Amazon MQ 會自動將版本中的所有代理程式更新到下一個支援的次要版本。Amazon MQ 至少會在引擎版本終止支援前 90 天發出通知。 若要檢視版本支援行事曆和終止支援，請參閱以下內容： • 管理 Amazon MQ for ActiveMQ 引擎版本 • 管理 Amazon MQ for RabbitMQ 引擎版本 您也可以啟用自動次要版本升級，讓代理程式在維護時段期間更新至下一個修補程式版本。如需詳細資訊，請參閱升級 Amazon MQ 代理程式引擎版本

Date	文件更新
2024 年 5 月 9 日	Amazon MQ for RabbitMQ 現在支援 RabbitMQ 3.12，這是一個次要版本版本。3.12.13 及更高版本的所有代理程式都使用 Classic Queues 第 2 版 (CQv2)， 3.12.13及更高版本的所有佇列都表現為延遲佇列。 我們建議在 3.12.13 之前的版本上啟用 CQv2 和延遲佇列的代理程式，或升級至最新版本的 Amazon MQ for RabbitMQ。 如需此版本中修正及功能的詳細資訊，請參閱下列項目： • RabbitMQ 伺服器 GitHub 儲存庫上的 RabbitMQ 3.12 版本備註。 RabbitMQ GitHub 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 3 月 4 日	Amazon MQ for RabbitMQ 現在支援 RabbitMQ 3.11.28。 如需此版本中修正及功能的詳細資訊，請參閱下列項目： • RabbitMQ 伺服器 GitHub 儲存庫上的 RabbitMQ 3.11.28 版本備註 RabbitMQ GitHub • RabbitMQ 變更記錄 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	Amazon MQ for RabbitMQ 不支援使用者名稱「訪客」，並且會在您建立新的代理程式時刪除預設訪客帳戶。Amazon MQ 也會定期刪除任何客戶建立的帳戶，稱為「訪客」。
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」。

Date	文件更新
2023 年 12 月 11 日	Amazon MQ for RabbitMQ 現在支援 RabbitMQ 3.10.25。 如需此版本中修正及功能的詳細資訊，請參閱下列項目： • RabbitMQ 伺服器 GitHub 儲存庫上的 RabbitMQ 3.10.25 版本備註 • RabbitMQ 變更記錄 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。 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。 如需此版本中修正及功能的詳細資訊，請參閱下列項目： • RabbitMQ 伺服器 GitHub 儲存庫上的 RabbitMQ 3.11.20 版本資訊 • RabbitMQ 變更記錄 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。

Date	文件更新
2023 年 7 月 27 日	Amazon MQ for RabbitMQ 現在支援 RabbitMQ 3.11.16。 如需此版本中修正及功能的詳細資訊，請參閱下列項目： • RabbitMQ 伺服器 GitHub 儲存庫上的 RabbitMQ 3.11.16 版本備註 • RabbitMQ 變更記錄 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。 如需新增組態至代理程式的詳細資訊，請參閱 RabbitMQ Broker Configurations。 如需有關此功能的詳細資訊，請參閱： • 操作員政策 • 操作員政策的變更
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。 如需詳細資訊，請參閱下列內容： • 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。 • ActiveMQ 5.17.3 發行頁面 • 管理 Amazon MQ for ActiveMQ 引擎版本 • 升級 Amazon MQ 代理程式引擎版本 • 使用 Spring XML 組態檔案

Date	文件更新
2023 年 6 月 21 日	Amazon MQ for ActiveMQ 現在提供跨區域資料複寫 (CRDR) 功能，允許從主要 AWS 區域中的主要代理程式非同步訊息複寫到複本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。 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： • 亞太地區 (墨爾本) • 亞太地區 (海德拉巴) • 歐洲 (西班牙) • 歐洲 (蘇黎世) 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 4 月 14 日	Amazon MQ for RabbitMQ 現在支援 RabbitMQ 3.9.27 版。 如需此版本中修正及功能的詳細資訊，請參閱下列項目： • RabbitMQ 伺服器 GitHub 儲存庫上的 RabbitMQ 3.9.27 版本備註 • RabbitMQ 變更記錄 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。

Date	文件更新
2023 年 4 月 14 日	Amazon MQ for RabbitMQ 現在支援 RabbitMQ 3.10.20 版。 如需此版本中修正及功能的詳細資訊，請參閱下列項目： • RabbitMQ 伺服器 GitHub 儲存庫上的 RabbitMQ 3.10.20 版本備註 • RabbitMQ 變更記錄 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 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 3.10.10 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。建議您啟用版本升級選項，以自動取得最新的錯誤修正、安全性更新和效能增強功能。 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。
2023 年 3 月 1 日	Amazon MQ for RabbitMQ 現在支援 RabbitMQ 3.10.17 版。 如需此版本中修正及功能的詳細資訊，請參閱下列項目： • RabbitMQ 伺服器 GitHub 儲存庫上的 RabbitMQ 3.10.17 版本備註 • RabbitMQ 變更記錄 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。

Date	文件更新
2023 年 2 月 21 日	Amazon MQ for RabbitMQ 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受管 CMK，或在 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。 Amazon MQ 支援以下列方式使用 AWS KMS 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。
2023 年 1 月 13 日	Amazon MQ for RabbitMQ 現在支援 RabbitMQ 3.8.34 版。 如需此版本中修正及功能的詳細資訊，請參閱下列項目： • RabbitMQ 伺服器 GitHub 儲存庫上的 RabbitMQ 3.8.34 版本備註 • RabbitMQ 變更記錄 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 15 日	Amazon MQ for RabbitMQ 現在支援 RabbitMQ 3.9.24 版。 如需此版本中修正及功能的詳細資訊，請參閱下列項目： • RabbitMQ 伺服器 GitHub 儲存庫上的 RabbitMQ 3.9.24 版本備註 • RabbitMQ 變更記錄 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」。

Date	文件更新
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： • RabbitMQ 3.10.10 版本備註 • RabbitMQ 變更記錄 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： • ActiveMQ 5.17.2 發行頁面 • 管理 Amazon MQ for ActiveMQ 引擎版本 • 升級 Amazon MQ 代理程式引擎版本 • 使用 Spring XML 組態檔案
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： • ActiveMQ 5.17.1 發行頁面 • 管理 Amazon MQ for ActiveMQ 引擎版本 • 升級 Amazon MQ 代理程式引擎版本 • 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： • ActiveMQ 5.16.5 發行頁面 • 管理 Amazon MQ for ActiveMQ 引擎版本 • 使用 Spring XML 組態檔案 • 升級 Amazon MQ 代理程式引擎版本

Date	文件更新
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 • 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。 • 疑難排解 • 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： • ActiveMQ 5.16.4 發行頁面 • 管理 Amazon MQ for ActiveMQ 引擎版本 • 使用 Spring XML 組態檔案 • 升級 Amazon MQ 代理程式引擎版本
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。 如需此版本中修正及功能的詳細資訊，請參閱下列項目： • RabbitMQ 伺服器 GitHub 儲存庫上的 RabbitMQ 3.8.27 版本備註 • RabbitMQ 變更記錄 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」。

Date	文件更新
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。 如需 RabbitMQ 3.9 中推出之新功能的詳細資訊，請參閱 GitHub 網站上的「版本 3.9.0 的版本備註頁面」。 **Note** 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。 如需此版本中修正及功能的詳細資訊，請參閱下列項目： • RabbitMQ 伺服器 GitHub 儲存庫上的 RabbitMQ 3.9.13 版本備註 • RabbitMQ 變更記錄 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 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。 如需詳細資訊，請參閱下列內容： • the section called “RabbitMQ 的指標”

Date	文件更新
2022 年 1 月 18 日	Amazon MQ for RabbitMQ 現在支援 RabbitMQ 3.8.26 版。 如需此版本中修正及功能的詳細資訊，請參閱下列項目： • RabbitMQ 伺服器 GitHub 儲存庫上的 RabbitMQ 3.8.26 版本備註 • RabbitMQ 變更記錄 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。 • the section called “ `RABBITMQ_MEMORY_ALARM` ”
2022 年 1 月 6 日	當您為 Amazon MQ for ActiveMQ 代理程式設定 CloudWatch Logs 時，Amazon MQ 支援在 IAM 資源型政策中使用 `aws:SourceArn` 和 `aws:SourceAccount` 全域條件內容鍵，以防止混淆代理問題。如需更多資訊，請參閱下列內容。 • 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。 • the section called “ActiveMQ 的指標”

Date	文件更新
2021 年 11 月 16 日	Amazon MQ for RabbitMQ 現在支援 RabbitMQ 3.8.23 版。 如需此版本中修正及功能的詳細資訊，請參閱下列項目： • RabbitMQ 伺服器 GitHub 儲存庫上的 RabbitMQ 3.8.23 版本備註 • RabbitMQ 變更記錄 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： • ActiveMQ 5.16.3 發行頁面 • 管理 Amazon MQ for ActiveMQ 引擎版本 • 升級 Amazon MQ 代理程式引擎版本 • 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。 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。 如需此版本中修正及功能的詳細資訊，請參閱下列項目： • RabbitMQ 伺服器 GitHub 儲存庫上的 RabbitMQ 3.8.22 版本備註 • RabbitMQ 變更記錄 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	Amazon MQ for RabbitMQ 已暫時停用 RabbitMQ 引擎 3.8.17 版，因為使用每則訊息存留時間 (TTL) 的佇列 發現問題。我們建議使用 3.8.11 版。

Date	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： • RabbitMQ 伺服器 GitHub 儲存庫上的 RabbitMQ 3.8.17 版本備註 • RabbitMQ 變更記錄 • 管理 Amazon MQ for RabbitMQ 引擎版本
2021 年 7 月 16 日	您現在可以使用 AWS 管理主控台 AWS CLI或 Amazon MQ API 來調整 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。 • 排程 Amazon MQ 代理程式的維護時段
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： • 一致雜湊交換外掛程式 • RabbitMQ GitHub 儲存庫上的 RabbitMQ 一致雜湊交換類型
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： • ActiveMQ 5.16.2 發行頁面 • 管理 Amazon MQ for ActiveMQ 引擎版本 • 升級 Amazon MQ 代理程式引擎版本 • 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。

Date	文件更新
2021 年 5 月 18 日	Amazon MQ for RabbitMQ 會實作代理程式預設值。 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： https://docs.aws.amazon.com//amazon-mq/latest/developer-guide/rabbitmq-defaults.html
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： • ActiveMQ 5.15.15 發行頁面 • 管理 Amazon MQ for ActiveMQ 引擎版本 • 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： • 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： • RabbitMQ 伺服器 GitHub 儲存庫上的 RabbitMQ 3.8.11 版本備註 • RabbitMQ 變更記錄 • 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。

Date	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： • ActiveMQ 5.15.14 版本備註 • 管理 Amazon MQ for ActiveMQ 引擎版本 • 使用 Spring XML 組態檔案 • **Important** 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。這可讓您將現有的 RabbitMQ 訊息代理程式遷移到 ， AWS 而無需重寫程式碼。 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。 • 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”。 • 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。 • 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”。 • 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。 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 入門：建立並連線至 RabbitMQ 代理程式。

Date	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： • ActiveMQ 5.15.13 版本備註 • 管理 Amazon MQ for ActiveMQ 引擎版本 • 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： • ActiveMQ 5.15.12 版本備註 • 管理 Amazon MQ for ActiveMQ 引擎版本 • 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。 Amazon MQ 也支援 `kahaDB` 子元素上的三個新屬性。 • `journalDiskSyncInterval` - 若為 `journalDiskSyncStrategy=periodic` ，要在何時執行磁碟同步的間隔 (毫秒)。 • `journalDiskSyncStrategy` - 設定磁碟同步政策。 • `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，請參閱屬性。

Date	文件更新
2020 年 3 月 3 日	Amazon MQ 支援兩個新的 CloudWatch 指標 • `TempPercentUsage` - 非持久性訊息使用的可用臨時儲存百分比。 • `JobSchedulerStorePercentUsage` - 任務排程器存放區使用的磁碟空間百分比。 如需詳細資訊，請參閱Monitoring and logging Amazon MQ brokers。
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： • ActiveMQ 5.15.10 版本備註 • 管理 Amazon MQ for ActiveMQ 引擎版本 • 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。

Date	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。 **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。 如需詳細資訊，請參閱下列內容： • Storage • 為最佳輸送量選擇正確的代理程式儲存類型 • Amazon MQ REST API 參考中 `broker-instance-options` 資源的 `storageType` 屬性 • Monitoring and logging Amazon MQ brokers 章節中的 `BurstBalance` 、`VolumeReadOps` 和 `VolumeWriteOps` 指標。
2019 年 10 月 18 日	現在提供 2 個 Amazon CloudWatch 指標：`TotalEnqueueCount` 和 `TotalDequeueCount` 。如需詳細資訊，請參閱Monitoring and logging Amazon MQ brokers

Date	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。 如需詳細資訊，請參閱下列內容： • 美國聯邦資訊處理標準 (FIPS) 140-2 • 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受管 CMK，或在 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。 Amazon MQ 支援以下列方式使用 AWS KMS 金鑰。 • AWS 擁有的 KMS 金鑰 — 金鑰為 Amazon MQ 擁有，不在您的帳戶中。 • AWS 受管 KMS 金鑰 — AWS 受管 KMS 金鑰 (aws/mq) 是帳戶中的 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 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。

Date	文件更新
2019 年 6 月 3 日	現在提供 2 個新的 Amazon CloudWatch 指標：`EstablishedConnectionsCount` 和 `InactiveDurableSubscribers` 。如需詳細資訊，請參閱下列內容： • Monitoring and logging Amazon MQ brokers • Monitoring and logging Amazon MQ brokers
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： • the section called “資料儲存體” • 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： • Amazon MQ 如何搭配 IAM 運作 • 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： • 代理程式執行個體選項 • 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： • ActiveMQ 5.15.9 版本備註 • 管理 Amazon MQ for ActiveMQ 引擎版本 • 使用 Spring XML 組態檔案

Date	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： • 搭配傳輸連接器的動態容錯移轉 • 代理程式的 Amazon MQ 網路 • 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： • 亞太地區 (新加坡) • 美國東部 (俄亥俄) • 美國東部 (維吉尼亞北部) • 美國西部 (加利佛尼亞北部) • 美國西部 (奧勒岡) • 亞太地區 (東京) • 亞太地區 (首爾) • 亞太地區 (悉尼) • 歐洲 (法蘭克福) • 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。

Date	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： • 代理程式的 Amazon MQ 網路 • Creating and Configuring a Network of Brokers • 正確地設定您的代理程式網路 • `networkConnector` • `networkConnectionStartAsync`

2018 年 12 月 11 日	Amazon MQ 支援 ActiveMQ 5.15.8、5.15.6 和 5.15.0。
	• ActiveMQ 文件中已解決的錯誤和改進項目：
	  • ActiveMQ 5.15.8 版本備註
	  • ActiveMQ 5.15.7 版本備註

2018 年 12 月 5 日	AWS 支援資源標記，以協助追蹤您的成本分配。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 已擴展其 SOC 合規計劃，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。 • 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 已擴展其 HIPAA 合規計劃，將 Amazon MQ 納入 HIPAA 合格服務。

Date	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：
	• 入門：建立並連線至 ActiveMQ 代理程式
	• ActiveMQ 文件中已解決的錯誤和改善：
	  • ActiveMQ 5.15.6 版本備註
	  • ActiveMQ 5.15.5 版本備註
	  • ActiveMQ 5.15.4 版本備註
	  • ActiveMQ 5.15.3 版本備註
	  • ActiveMQ 5.15.2 版本備註
	  • ActiveMQ 5.15.1 版本備註
	• ActiveMQ Client 5.15.6

2018 年 8 月 31 日	• 下列指標可供使用：
	  • `CurrentConnectionsCount`
	  • `TotalConsumerCount`
	  • `TotalProducerCount`
	  如需詳細資訊，請參閱 Monitoring and logging Amazon MQ brokers 一節。
	• 代理程式的 IP 地址會顯示在詳細資訊頁面上。
	  Note
	  對於公開存取性已停用的代理程式，會顯示內部 IP 地址。

Date	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： • 美國東部 (俄亥俄) • 美國東部 (維吉尼亞北部) • 美國西部 (加利佛尼亞北部) • 美國西部 (奧勒岡) • 亞太地區 (東京) • 亞太地區 (首爾) • 亞太地區 (悉尼) • 歐洲 (法蘭克福) • 歐洲 (愛爾蘭)
2018 年 7 月 30 日	您可以設定 Amazon MQ，將一般和稽核日誌發佈至 Amazon CloudWatch Logs。如需詳細資訊，請參閱Monitoring and logging Amazon MQ brokers。
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： • 美國東部 (俄亥俄) • 美國東部 (維吉尼亞北部) • 美國西部 (加利佛尼亞北部) • 美國西部 (奧勒岡) • 亞太地區 (悉尼) • 歐洲 (法蘭克福) • 歐洲 (愛爾蘭)
2018 年 7 月 19 日	您可以使用 AWS CloudTrail 記錄 Amazon MQ API 呼叫。如需詳細資訊，請參閱Logging Amazon MQ API calls using CloudTrail。

Date	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： • `mq.m5.large` • `mq.m5.xlarge` • `mq.m5.2xlarge` • `mq.m5.4xlarge` 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： • 美國東部 (俄亥俄) • 美國東部 (維吉尼亞北部) • 美國西部 (奧勒岡) • 亞太地區 (悉尼) • 歐洲 (法蘭克福) • 歐洲 (愛爾蘭)

Date	文件更新

2018 年 6 月 14 日	• 您可以使用 `AWS::Amazon MQ::Broker` AWS CloudFormation 資源來執行下列動作：
	  • 建立代理程式。
	  • 新增組態變更或修改所指定代理程式的使用者。
	  • 傳回所指定代理程式的相關資訊。
	  • 刪除指定的代理程式。
	
	  Note
	  變更 Amazon MQ Broker ConfigurationId (Amazon MQ 代理程式 ConfigurationId) 或 Amazon MQ Broker User (Amazon MQ 代理程式使用者) 屬性類別的任何屬性時，代理程式會立即重新啟動。
	• 您可以使用 `AWS::Amazon MQ::Configuration` AWS CloudFormation 資源來執行下列動作：
	  • 建立組態。
	  • 更新指定的組態。
	  • 傳回所指定組態的相關資訊。
	
	  Note
	  您可以使用 CloudFormation 修改，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。

Date	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。 • `>CpuCreditBalance` CloudWatch 指標可用於`mq.t2.micro` 代理程式執行個體類型。
2018 年 1 月 10 日	以下變更會影響 Amazon MQ 主控台： • 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  。 • 在 ***MyBroker*** 頁面的連線區段中，選擇安全群組的名稱或  會開啟 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Connecting a Java application to your broker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` 。 • 錯誤的 `mq:DescribeEngine` 許可會從 IAM 主控台中移除。

Date	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。
	• Amazon MQ 已在下列區域提供 ：
	  • 美國東部 (俄亥俄)
	  • 美國東部 (維吉尼亞北部)
	  • 美國西部 (奧勒岡)
	  • 亞太地區 (悉尼)
	  • 歐洲 (法蘭克福)
	  • 歐洲 (愛爾蘭)
	  `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。
	• 您可以建立 `mq.m4.large` 和 `mq.t2.micro` 代理程式。
	  `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。
	• 您可以使用 ActiveMQ 5.15.0 代理程式引擎。
	• 您也可以使用 Amazon MQ REST API 和 AWS SDKs，以程式設計方式建立和管理代理程式。
	• 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

Date	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。 • 您的代理程式可以傳送和接收任何大小的訊息。

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。