Amazon MQ: Developer Guide
Copyright © 2020 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>What is Amazon MQ?</td>
<td>1</td>
</tr>
<tr>
<td>What Are the Main Benefits of Amazon MQ?</td>
<td>1</td>
</tr>
<tr>
<td>How Is Amazon MQ Different from Amazon SQS or Amazon SNS?</td>
<td>1</td>
</tr>
<tr>
<td>How Can I Get Started with Amazon MQ?</td>
<td>2</td>
</tr>
<tr>
<td>We Want to Hear from You</td>
<td>2</td>
</tr>
<tr>
<td>Setting Up</td>
<td>3</td>
</tr>
<tr>
<td>Step 1: Create an AWS Account and an IAM Administrator User</td>
<td>3</td>
</tr>
<tr>
<td>Step 2: Create an IAM User and Get Your AWS Credentials</td>
<td>3</td>
</tr>
<tr>
<td>Step 3: Get Ready to Use the Example Code</td>
<td>4</td>
</tr>
<tr>
<td>Next Steps</td>
<td>4</td>
</tr>
<tr>
<td>Getting Started</td>
<td>5</td>
</tr>
<tr>
<td>Prerequisites</td>
<td>5</td>
</tr>
<tr>
<td>Step 1: Create an ActiveMQ Broker</td>
<td>5</td>
</tr>
<tr>
<td>Step 2: Connect a Java Application to Your Broker</td>
<td>6</td>
</tr>
<tr>
<td>Prerequisites</td>
<td>6</td>
</tr>
<tr>
<td>Create a Message Producer and Send a Message</td>
<td>7</td>
</tr>
<tr>
<td>Create a Message Consumer and Receive the Message</td>
<td>8</td>
</tr>
<tr>
<td>Step 3: Delete Your Broker</td>
<td>9</td>
</tr>
<tr>
<td>Next Steps</td>
<td>10</td>
</tr>
<tr>
<td>Tutorials</td>
<td>11</td>
</tr>
<tr>
<td>Creating and Configuring a Broker</td>
<td>11</td>
</tr>
<tr>
<td>Step 1: Configure Basic Broker Settings</td>
<td>11</td>
</tr>
<tr>
<td>Step 2: (Optional) Configure Additional Broker Settings</td>
<td>12</td>
</tr>
<tr>
<td>Step 3: Finish Creating the Broker</td>
<td>13</td>
</tr>
<tr>
<td>Accessing the ActiveMQ Web Console of a Broker without Public Accessibility</td>
<td>14</td>
</tr>
<tr>
<td>Creating and Configuring a Network of Brokers</td>
<td>15</td>
</tr>
<tr>
<td>Prerequisites</td>
<td>16</td>
</tr>
<tr>
<td>Step 1: Allow Traffic between Brokers</td>
<td>16</td>
</tr>
<tr>
<td>Step 2: Configure Network Connectors for Your Broker</td>
<td>17</td>
</tr>
<tr>
<td>Next Steps</td>
<td>18</td>
</tr>
<tr>
<td>Editing Broker Preferences</td>
<td>19</td>
</tr>
<tr>
<td>To Edit Broker Options</td>
<td>19</td>
</tr>
<tr>
<td>Creating and Applying Configurations</td>
<td>20</td>
</tr>
<tr>
<td>Step 1: Create a Configuration from Scratch</td>
<td>20</td>
</tr>
<tr>
<td>Step 2: Create a New Configuration Revision</td>
<td>21</td>
</tr>
<tr>
<td>Step 3: Apply a Configuration Revision to Your Broker</td>
<td>21</td>
</tr>
<tr>
<td>Editing Configurations and Managing Configuration Revisions</td>
<td>22</td>
</tr>
<tr>
<td>To View a Previous Configuration Revision</td>
<td>23</td>
</tr>
<tr>
<td>To Edit the Current Configuration Revision</td>
<td>23</td>
</tr>
<tr>
<td>To Apply a Configuration Revision to Your Broker</td>
<td>24</td>
</tr>
<tr>
<td>To Roll Back Your Broker to the Last Configuration Revision</td>
<td>24</td>
</tr>
<tr>
<td>Connecting a Java Application to Your Broker</td>
<td>25</td>
</tr>
<tr>
<td>Prerequisites</td>
<td>25</td>
</tr>
<tr>
<td>To Create a Message Producer and Send a Message</td>
<td>26</td>
</tr>
<tr>
<td>To Create a Message Consumer and Receive the Message</td>
<td>28</td>
</tr>
<tr>
<td>Listing Brokers and Viewing Broker Details</td>
<td>29</td>
</tr>
<tr>
<td>To List Brokers and View Broker Details</td>
<td>29</td>
</tr>
<tr>
<td>Creating and Managing Broker Users</td>
<td>30</td>
</tr>
<tr>
<td>To Create a New User</td>
<td>31</td>
</tr>
<tr>
<td>To edit an existing user</td>
<td>31</td>
</tr>
<tr>
<td>To Delete an Existing User</td>
<td>32</td>
</tr>
<tr>
<td>Rebooting a Broker</td>
<td>32</td>
</tr>
<tr>
<td>To Reboot an Amazon MQ Broker</td>
<td>32</td>
</tr>
<tr>
<td>Deleting a Broker</td>
<td>33</td>
</tr>
</tbody>
</table>
What Is Amazon MQ?

Amazon MQ is a managed message broker service for Apache ActiveMQ that makes it easy to migrate to a message broker in the cloud. A message broker allows software applications and components to communicate using various programming languages, operating systems, and formal messaging protocols.

Amazon MQ works with your existing applications and services without the need to manage, operate, or maintain your own messaging system.

Topics
- What Are the Main Benefits of Amazon MQ? (p. 1)
- How Is Amazon MQ Different from Amazon SQS or Amazon SNS? (p. 1)
- How Can I Get Started with Amazon MQ? (p. 2)
- We Want to Hear from You (p. 2)

What Are the Main Benefits of Amazon MQ?

- **Security** – You control who can create and modify brokers (p. 117) and who can send messages to and receive messages from (p. 120) an ActiveMQ broker. Amazon MQ encrypts messages at rest and in transit using encryption keys that it manages and stores securely.
- **Durability** – To ensure the safety of your messages, Amazon MQ stores them on redundant shared storage (p. 44).
- **Availability** – You can create a single-instance broker (p. 46) (comprised of one broker in one Availability Zone), or an active/standby broker for high availability (p. 49) (comprised of two brokers in two different Availability Zones). For either broker type, Amazon MQ automatically provisions infrastructure for high durability.
- **Compatibility** – Amazon MQ supports industry-standard APIs and protocols so you can migrate from your existing message broker (p. 88) without rewriting application code (p. 78).
- **Operation offloading** – You can configure many aspects of your ActiveMQ broker (p. 61), such as predefined destinations, destination policies, authorization policies, and plugins. Amazon MQ controls some of these configuration elements, such as network transports and storage, simplifying the maintenance and administration of your messaging system in the cloud.

How Is Amazon MQ Different from Amazon SQS or Amazon SNS?

Amazon MQ is a managed message broker service that provides compatibility with many popular message brokers. We recommend Amazon MQ for migrating applications from existing message brokers that rely on compatibility with APIs such as JMS or protocols such as AMQP, MQTT, OpenWire, and STOMP.

Amazon SQS and Amazon SNS are queue and topic services that are highly scalable, simple to use, and don't require you to set up message brokers. We recommend these services for new applications that can benefit from nearly unlimited scalability and simple APIs.
How Can I Get Started with Amazon MQ?

- To create your first broker with Amazon MQ, see Getting Started with Amazon MQ (p. 5).
- To discover the functionality and architecture of Amazon MQ, see How Amazon MQ Works (p. 37).
- To find out the guidelines and caveats that will help you make the most of Amazon MQ, see Best Practices for Amazon MQ (p. 93).
- To learn about Amazon MQ REST APIs, see the Amazon MQ REST API Reference.
- To learn about Amazon MQ AWS CLI commands, see Amazon MQ in the AWS CLI Command Reference.

We Want to Hear from You

We welcome your feedback. To contact us, visit the Amazon MQ Discussion Forum.
Setting Up Amazon MQ

Before you can use Amazon MQ, you must complete the following steps.

Topics
- Step 1: Create an AWS Account and an IAM Administrator User (p. 3)
- Step 2: Create an IAM User and Get Your AWS Credentials (p. 3)
- Step 3: Get Ready to Use the Example Code (p. 4)
- Next Steps (p. 4)

Step 1: Create an AWS Account and an IAM Administrator User

To access any AWS service, you must first create an AWS account. This is an Amazon account that can use AWS products. You can use your AWS account to view your activity and usage reports and to manage authentication and access.

1. Navigate to the AWS home page, and then choose Create an AWS Account.
2. Follow the instructions.
   - Part of the sign-up procedure involves receiving a phone call and entering a PIN using the phone keypad.
3. When you finish creating your AWS account, follow the instructions in the IAM User Guide to create your first IAM administrator user and group.

Step 2: Create an IAM User and Get Your AWS Credentials

To avoid using your IAM administrator user for Amazon MQ operations, it is a best practice to create an IAM user for each person who needs administrative access to Amazon MQ.

To work with Amazon MQ, you need the AmazonMQFullAccess policy and AWS credentials that are associated with your IAM user. These credentials are comprised of an access key ID and a secret access key. For more information, see What Is IAM? in the IAM User Guide and AWS Security Credentials in the AWS General Reference.

1. Sign in to the AWS Identity and Access Management console.
2. Choose Users, Add user.
3. Type a User name, such as AmazonMQAdmin.
4. Select Programmatic access and AWS Management Console access.
5. Set a Console password and then choose Next: Permissions.
6. On the Set permissions for AmazonMQAdmin page, choose Attach existing policies directly.
7. Type AmazonMQ into the filter, choose AmazonMQFullAccess, and then choose Next: Review.
8. On the Review page, choose Create user.
Step 3: Get Ready to Use the Example Code

The following tutorials show how you can work with Amazon MQ and ActiveMQ using the AWS Management Console and Java. To use the example code, you must install the Java Standard Edition Development Kit and make some changes to the code.

You can also create and manage brokers programmatically using Amazon MQ REST API and AWS SDKs.

Next Steps

Now that you're prepared to work with Amazon MQ, get started by creating a broker (p. 5) and then connecting a Java application (p. 25) to your broker.

You can also try the more advanced Amazon MQ tutorials (p. 11).

For more information on configuring a network of brokers, see Network of Brokers (p. 50).

The IAM user is created and the Access key ID is displayed, for example:

AKIAIOSFODNN7EXAMPLE

9. To display your Secret access key, choose Show, for example:

wJalrXUtneFEMI/K7MDENG/bPxFiCvYEXAMPLEKEY

Important
You can view or download your secret access key only when you create your credentials (however, you can create new credentials at any time).

10. To download your credentials, choose Download .csv. Keep this file in a secure location.
Getting Started with Amazon MQ

This section will help you become more familiar with Amazon MQ by showing you how to create a broker and how to connect your application to it.

The following 3-minute video provides a preview of creating and using an Amazon MQ broker.

Topics
- Prerequisites (p. 5)
- Step 1: Create an ActiveMQ Broker (p. 5)
- Step 2: Connect a Java Application to Your Broker (p. 6)
- Step 3: Delete Your Broker (p. 9)
- Next Steps (p. 10)

Prerequisites

Before you begin, complete the steps in Setting Up Amazon MQ (p. 3).

Step 1: Create an ActiveMQ Broker

A broker is a message broker environment running on Amazon MQ. It is the basic building block of Amazon MQ. The combined description of the broker instance class (m5, t2) and size (large, micro) is a broker instance type (for example, mq.m5.large). For more information, see Broker (p. 37).

The first and most common Amazon MQ task is creating a broker. The following example shows how you can use the AWS Management Console to create a basic broker.

1. Sign in to the Amazon MQ console.
2. On the Select deployment and storage page, in the Deployment mode and storage type section, do the following:
   a. Choose the Deployment mode (for example, Active/standby broker). For more information, see Amazon MQ Broker Architecture (p. 45).
      - A Single-instance broker is comprised of one broker in one Availability Zone. The broker communicates with your application and with Amazon EFS (by default) or with Amazon EBS. For more information, see Amazon MQ Single-Instance Broker (p. 46).
      - An Active/standby broker for high availability is comprised of two brokers in two different Availability Zones, configured in a redundant pair. These brokers communicate synchronously with your application, and with Amazon EFS. For more information, see Amazon MQ Active/Standby Broker for High Availability (p. 49).
      - For more information on the sample blueprints for a network of brokers, see Sample Blueprints (p. 52).
   b. Choose the Storage type (for example, EBS). For more information, see Broker Storage (p. 44).
      Note
      Amazon EBS replicates data within a single Availability Zone and doesn’t support the active/standby (p. 49) deployment mode.
   c. Choose Next.
3. On the Configure settings page, in the Details section, do the following:
Step 2: Connect a Java Application to Your Broker

After you create an Amazon MQ broker, you can connect your application to it. The following examples show how you can use the Java Message Service (JMS) to create a connection to the broker, create a queue, and send a message. For a complete, working Java example, see Working Java Example (p. 78).

You can connect to ActiveMQ brokers using various ActiveMQ clients. We recommend using the ActiveMQ Client.

**Prerequisites**

**Enable VPC Attributes**

To ensure that your broker is accessible within your VPC, you must enable the enableDnsHostnames and enableDnsSupport VPC attributes. For more information, see DNS Support in your VPC in the Amazon VPC User Guide.

**Enable Inbound Connections**

1. Sign in to the Amazon MQ console.
2. From the broker list, choose the name of your broker (for example, MyBroker).
3. On the MyBroker page, in the Connections section, note the addresses and ports of the broker's ActiveMQ Web Console URL and wire-level protocols.

4. In the Details section, under Security and network, choose the name of your security group or [View Security Group]

The Security Groups page of the EC2 Dashboard is displayed.

5. From the security group list, choose your security group.

6. At the bottom of the page, choose Inbound, and then choose Edit.

7. In the Edit inbound rules dialog box, add a rule for every URL or endpoint that you want to be publicly accessible (the following example shows how to do this for an ActiveMQ Web Console).

   a. Choose Add Rule.
   b. For Type, select Custom TCP.
   c. For Port Range, type the ActiveMQ Web Console port (8162).
   d. For Source, leave Custom selected and then type the IP address of the system that you want to be able to access the ActiveMQ Web Console (for example, 192.0.2.1).
   e. Choose Save.

Your broker can now accept inbound connections.

Add Java Dependencies

Add the activemq-client.jar and activemq-pool.jar packages to your Java class path. The following example shows these dependencies in a Maven project pom.xml file.

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

For more information about activemq-client.jar, see Initial Configuration in the Apache ActiveMQ documentation.

Important

In the following example code, producers and consumers run in a single thread. For production systems (or to test broker instance failover), make sure that your producers and consumers run on separate hosts or threads.

Create a Message Producer and Send a Message

1. Create a JMS pooled connection factory for the message producer using your broker's endpoint and then call the createConnection method against the factory.

   Note
   For an active/standby broker, Amazon MQ provides two ActiveMQ Web Console URLs, but only one URL is active at a time. Likewise, Amazon MQ provides two endpoints for each wire-level protocol, but only one endpoint is active in each pair at a time. The -1 and -2 suffixes denote a redundant pair. For more information, see Amazon MQ Broker Architecture (p. 45)).
For wire-level protocol endpoints, you can allow your application to connect to either endpoint by using the Failover Transport.

```java
// Create a connection factory.
final ActiveMQConnectionFactory connectionFactory = new ActiveMQConnectionFactory(wireLevelEndpoint);

// Pass the username and password.
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();
```

**Note**
Message producers should always use the PooledConnectionFactory class. For more information, see Always Use Connection Pooling (p. 95).

2. Create a session, a queue named `MyQueue`, and a message producer.

```java
// 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. Create the message string "Hello from Amazon MQ!" and then send the message.

```java
// 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. Clean up the producer.

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

---

Create a Message Consumer and Receive the Message

1. Create a JMS connection factory for the message producer using your broker's endpoint and then call the createConnection method against the factory.

```java
// Create a connection factory.
```
final ActiveMQConnectionFactory connectionFactory = new ActiveMQConnectionFactory(wireLevelEndpoint);

// Pass the username and password.
connectionFactory.setUserName(activeMqUsername);
connectionFactory.setPassword(activeMqPassword);

// Establish a connection for the consumer.
final Connection consumerConnection = connectionFactory.createConnection();
consumerConnection.start();

Note
Message consumers should never use the PooledConnectionFactory class. For more information, see Always Use Connection Pooling (p. 95).

2. Create a session, a queue named MyQueue, and a message consumer.

// 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. Begin to wait for messages and receive the message when it arrives.

// 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
Unlike AWS messaging services (such as Amazon SQS), the consumer is constantly connected to the broker.

4. Close the consumer, session, and connection.

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

Step 3: Delete Your Broker

If you don’t use an Amazon MQ broker (and don’t foresee using it in the near future), it is a best practice to delete it from Amazon MQ to reduce your AWS costs.

The following example shows how you can delete a broker using the AWS Management Console.

1. Sign in to the Amazon MQ console.
2. From the broker list, select your broker (for example, MyBroker) and then choose Delete.
3. In the Delete MyBroker? dialog box, type delete and then choose Delete.

Deleting a broker takes about 5 minutes.
Next Steps

Now that you have created a broker, connected an application to it, and sent and received a message, you might want to try the following:

- Creating and Configuring a Broker (p. 11) (Additional Settings)
- Editing Broker Engine Version, CloudWatch Logs, and Maintenance Preferences (p. 19)
- Creating and Applying Broker Configurations (p. 20)
- Editing and Managing Broker Configurations (p. 22)
- Listing Brokers and Viewing Broker Details (p. 29)
- Creating and Managing Amazon MQ Broker Users (p. 30)
- Rebooting a Broker (p. 32)
- Accessing CloudWatch Metrics for Amazon MQ (p. 33)

You can also begin to dive deep into Amazon MQ best practices (p. 93) and Amazon MQ REST APIs, and then plan to migrate to Amazon MQ (p. 88).
Amazon MQ Tutorials

The following tutorials show how you can work with Amazon MQ and ActiveMQ using the AWS Management Console and Java. To use the example code, you must install the Java Standard Edition Development Kit and make some changes to the code.

Topics
- Tutorial: Creating and Configuring an Amazon MQ Broker (p. 11)
- Tutorial: Creating and Configuring an Amazon MQ Network of Brokers (p. 15)
- Tutorial: Editing Broker Engine Version, Instance Type, CloudWatch Logs, and Maintenance Preferences (p. 19)
- Tutorial: Creating and Applying Amazon MQ Broker Configurations (p. 20)
- Tutorial: Editing Amazon MQ Broker Configurations and Managing Configuration Revisions (p. 22)
- Tutorial: Connecting a Java Application to Your Amazon MQ Broker (p. 25)
- Tutorial: Listing Amazon MQ Brokers and Viewing Broker Details (p. 29)
- Tutorial: Creating and Managing Amazon MQ Broker Users (p. 30)
- Tutorial: Rebooting an Amazon MQ Broker (p. 32)
- Tutorial: Deleting an Amazon MQ Broker (p. 33)
- Tutorial: Accessing CloudWatch Metrics for Amazon MQ (p. 33)

Tutorial: Creating and Configuring an Amazon MQ Broker

A broker is a message broker environment running on Amazon MQ. It is the basic building block of Amazon MQ. The combined description of the broker instance class (m5, t2) and size (large, micro) is a broker instance type (for example, mq.m5.large). For more information, see Broker (p. 37).

The first and most common Amazon MQ task is creating a broker. The following example shows how you can use the AWS Management Console to create and configure a broker using the AWS Management Console.

Topics
- Step 1: Configure Basic Broker Settings (p. 11)
- Step 2: (Optional) Configure Additional Broker Settings (p. 12)
- Step 3: Finish Creating the Broker (p. 13)
- Accessing the ActiveMQ Web Console of a Broker without Public Accessibility (p. 14)

Step 1: Configure Basic Broker Settings

1. Sign in to the Amazon MQ console.
2. On the Select deployment and storage page, in the Deployment mode and storage type section, do the following:
   a. Choose the Deployment mode (for example, Active/standby broker). For more information, see Amazon MQ Broker Architecture (p. 45).
Step 2: (Optional) Configure Additional Broker Settings

- **A Single-instance broker** is comprised of one broker in one Availability Zone. The broker communicates with your application and with Amazon EFS (by default) or with Amazon EBS. For more information, see Amazon MQ Single-Instance Broker (p. 46).
- **An Active/standby broker for high availability** is comprised of two brokers in two different Availability Zones, configured in a redundant pair. These brokers communicate synchronously with your application, and with Amazon EFS. For more information, see Amazon MQ Active/Standby Broker for High Availability (p. 49).
- For more information on the sample blueprints for a network of brokers, see Sample Blueprints (p. 52).

b. Choose the **Storage type** (for example, EBS). For more information, see Broker Storage (p. 44).
   
   **Note**
   Amazon EBS replicates data within a single Availability Zone and doesn't support the active/standby (p. 49) deployment mode.

c. Choose **Next**.

3. On the **Configure settings** page, in the **Details** section, do the following:

   a. Enter the **Broker name**.
   
   b. Choose the **Broker instance type** (for example, mq.m5.large). For more information, see Broker Instance Types (p. 39).

4. In the **ActiveMQ Web Console access** section, type a **Username** and **Password**.

**Step 2: (Optional) Configure Additional Broker Settings**

- **Important**
  
  - **Subnet(s)** – A single-instance broker requires one subnet (for example, the default subnet). An active/standby broker requires two subnets.
  
  - **Security group(s)** – Both single-instance brokers and active/standby brokers require at least one security group (for example, the default security group).
  
  - **VPC** – A broker's subnet(s) and security group(s) must be in the same VPC. EC2-Classic resources aren't supported. Amazon MQ only supports default VPC tenancy, and does not support dedicated VPC tenancy.
  
  - **Encryption** – Choose the customer master key to encrypt your data. See Encryption at Rest (p. 116).
  
  - **Public accessibility** – Disabling public accessibility makes the broker accessible only within your VPC. For more information, see Prefer Brokers without Public Accessibility (p. 93) and Accessing the ActiveMQ Web Console of a Broker without Public Accessibility (p. 14).

1. Expand the **Additional settings** section.
2. In the **Configuration** section, choose **Create a new configuration with default values** or **Select an existing configuration**. For more information, see Configuration (p. 42) and Amazon MQ Broker Configuration Parameters (p. 61).
3. In the **Logs** section, choose whether to publish **General** logs and **Audit** logs to Amazon CloudWatch Logs. For more information, see Configuring Amazon MQ to Publish Logs to Amazon CloudWatch Logs (p. 110).

   **Important**
   If you don't add the CreateLogGroup permission to your Amazon MQ user (p. 111) before the user creates or reboots the broker, Amazon MQ doesn't create the log group.
If you don’t configure a resource-based policy for Amazon MQ (p. 112), the broker can’t publish the logs to CloudWatch Logs.

4. In the **Network and security section**, configure your broker’s connectivity:
   a. Do one of the following:
      - Choose **Use the default VPC, subnet(s), and security group(s)**.
      - Choose **Select existing VPC, subnet(s), and security group(s)**.
        1. If you choose this option, you can create a new **Virtual Private Cloud (VPC)** on the Amazon VPC console, select an existing VPC, or select the default VPC. For more information, see *What is Amazon VPC?* in the *Amazon VPC User Guide*.
        2. After you create or select a VPC, you can create new **Subnet(s)** on the Amazon VPC console or select existing ones. For more information, see *VPCs and Subnets* in the *Amazon VPC User Guide*.
        3. After you create or select subnets, you can select the **Security group(s)**.
   b. Choose the **Public accessibility** of your broker.

5. In the **Maintenance section**, configure your broker’s maintenance schedule:
   a. To upgrade the broker to new versions as Apache releases them, choose **Enable automatic minor version upgrades**. Automatic upgrades occur during the *maintenance window* defined by the day of the week, the time of day (in 24-hour format), and the time zone (UTC by default).
      
      **Note**
      
      For an active/standby broker, if one of the broker instances undergoes maintenance, it takes Amazon MQ a short while to take the inactive instance out of service. This allows the healthy standby instance to become active and to begin accepting incoming communications.
   b. Do one of the following:
      - To allow Amazon MQ to select the maintenance window automatically, choose **No preference**.
      - To set a custom maintenance window, choose **Select maintenance window** and then specify the **Start day** and **Start time** of the upgrades.

### Step 3: Finish Creating the Broker

1. Choose **Deploy**.

   While Amazon MQ creates your broker, it displays the *Creation in progress* status.

   Creating the broker takes about 15 minutes.

   When your broker is created successfully, Amazon MQ displays the *Running* status.

   ![Broker Status](image)

   - **Name**: MyBroker
   - **Status**: Running
   - **Deployment mode**: Single-Instance broker
   - **Instance type**: mq.m5.large

2. Choose **MyBroker**.
On the **MyBroker** page, in the Connect section, note your broker’s **ActiveMQ Web Console** URL, for example:

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

Also, note your broker’s **wire-level protocol Endpoints**. The following is an example of an OpenWire endpoint:

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

**Note**
For an active/standby broker, Amazon MQ provides two ActiveMQ Web Console URLs, but only one URL is active at a time. Likewise, Amazon MQ provides two endpoints for each wire-level protocol, but only one endpoint is active in each pair at a time. The –1 and –2 suffixes denote a redundant pair. For more information, see Amazon MQ Broker Architecture (p. 45). For wire-level protocol endpoints, you can allow your application to connect to either endpoint by using the Failover Transport.

### Accessing the ActiveMQ Web Console of a Broker without Public Accessibility

If you disable public accessibility for your broker, you must perform the following steps to be able to access your broker’s ActiveMQ Web Console.

**Note**
The names of the VPCs and security groups are specific to the following example.

### Prerequisites

To perform the following steps, you must configure the following:

- **VPCs**
  - The VPC without an internet gateway, to which the Amazon MQ broker is attached, named `private-vpc`.
  - A second VPC, with an internet gateway, named `public-vpc`.
  - Both VPCs must be connected (for example, using VPC peering) so that the Amazon EC2 instances in the public VPC can communicate with the EC2 instances in the private VPC.
  - If you use VPC peering, the route tables for both VPCs must be configured for the peering connection.

- **Security Groups**
  - The security group used to create the Amazon MQ broker, named `private-sg`.
  - A second security group used for the EC2 instance in the `public-vpc` VPC, named `public-sg`.
  - `private-sg` must allow inbound connections from `public-sg`. We recommend restricting this security group to port 8162.
  - `public-sg` must allow inbound connections from your machine on port 22.

### To Access the ActiveMQ Web Console of a Broker without Public Accessibility

1. Create a Linux EC2 instance in `public-vpc` (with a public IP, if necessary).
2. To verify that your VPC is configured correctly, establish an ssh connection to the EC2 instance and use the curl command with the URI of your broker.

3. From your machine, create an ssh tunnel to the EC2 instance using the path to your private key file and the IP address of your broker instance. For example:

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

A forward proxy server is started on your machine.

4. Install a proxy client such as FoxyProxy on your machine.

5. Configure your proxy client using the following settings:
   - For proxy type, specify SOCKS5.
   - For IP address, DNS name, and server name, specify localhost.
   - For port, specify 8080.
   - Remove any existing URL patterns.
   - For the URL pattern, specify *.mq.*.amazonaws.com*
   - For the connection type, specify HTTP(S).

When you enable your proxy client, you can access the ActiveMQ Web Console on your machine.

### Tutorial: Creating and Configuring an Amazon MQ Network of Brokers

A network of brokers is comprised of multiple simultaneously active single-instance brokers (p. 46) or active/standby brokers (p. 49). You can configure networks of brokers in a variety of topologies (p. 52) (for example, concentrator, hub-and-spokes, tree, or mesh), depending on your application's needs, such as high availability and scalability. For instance, a hub and spoke (p. 55) network of brokers can increase resiliency, preserving messages if one broker is not reachable. A network of brokers with a concentrator (p. 56) topology can collect messages from a larger number of brokers accepting incoming messages, and concentrate them to more central brokers, to better handle the load of many incoming messages. In this tutorial, you learn how to create a two-broker network of brokers with a source and sink topology.

For a conceptual overview and detailed configuration information, see the following:

- **Amazon MQ Network of Brokers** (p. 50)
- **Configure Your Network of Brokers Correctly** (p. 98)
- networkConnector (p. 75)
- networkConnectionStartAsync (p. 71)
- **Networks of Brokers** in the ActiveMQ documentation

You can use the Amazon MQ console to create an Amazon MQ network of brokers. Because you can start the creation of the two brokers in parallel, this process takes approximately 15 minutes.

### Topics

- **Prerequisites** (p. 16)
- **Step 1: Allow Traffic between Brokers** (p. 16)
- **Step 2: Configure Network Connectors for Your Broker** (p. 17)
Prerequisites

To create a network of brokers, you must have the following:

- Two or more simultaneously active brokers (named MyBroker1 and MyBroker2 in this tutorial). For more information about creating brokers, see Creating and Configuring a Broker (p. 11).
- The two brokers must be in the same VPC or in peered VPCs. For more information about VPCs, see What is Amazon VPC? in the Amazon VPC User Guide and What is VPC Peering? in the Amazon VPC Peering Guide.

**Important**
If you don't have a default VPC, subnet(s), or security group, you must create them first. For more information, see the following in the Amazon VPC User Guide:

- Creating a Default VPC
- Creating a Default Subnet
- Creating a Security Group

- Two users with identical usernames and passwords for both brokers. For more information about creating users, see Creating and Managing Amazon MQ Broker Users (p. 30).

The following example uses two single-instance brokers (p. 46). However, you can create networks of brokers using active/standby brokers (p. 49) or a combination of broker deployment modes.

Step 1: Allow Traffic between Brokers

After you create your brokers, you must allow traffic between them.

1. On the Amazon MQ console, on the MyBroker2 page, in the Details section, under Security and network, choose the name of your security group or .

   The Security Groups page of the EC2 Dashboard is displayed.

2. From the security group list, choose your security group.

3. At the bottom of the page, choose Inbound, and then choose Edit.

4. In the Edit inbound rules dialog box, add a rule for the OpenWire endpoint.

   a. Choose Add Rule.
   b. For Type, select Custom TCP.
   c. For Port Range, type the OpenWire port (61617).
   d. Do one of the following:

      - If you want to restrict access to a particular IP address, for Source, leave Custom selected, and then enter the IP address of MyBroker1, followed by /32. (This converts the IP address to a valid CIDR record). For more information see Elastic Network Interfaces.

      **Tip**
      To retrieve the IP address of MyBroker1, on the Amazon MQ console, choose the name of the broker and navigate to the Details section.

      - If all the brokers are private and belong to the same VPC, for Source, leave Custom selected and then type the ID of the security group you are editing.

      **Note**
      For public brokers, you must restrict access using IP addresses.

   e. Choose Save.
Step 2: Configure Network Connectors for Your Broker

After you allow traffic between your brokers, you must configure network connectors for one of them.

1. Edit the configuration revision for broker *MyBroker1*.
   a. On the *MyBroker1* page, choose *Edit*.
   b. On the *Edit MyBroker1* page, in the *Configuration* section, choose *View*.

      The broker engine type and version that the configuration uses (for example, *Apache ActiveMQ 5.15.0*) are displayed.
   c. On the *Configuration details* tab, the configuration revision number, description, and broker configuration in XML format are displayed.
   d. Choose *Edit configuration*.
   e. At the bottom of the configuration file, uncomment the `<networkConnectors>` section and include the following information:

      - The name for the network connector.
      - *The ActiveMQ Web Console username* (p. 16) that is common to both brokers.
      - Enable duplex connections.
      - Do one of the following:

         - If you are connecting the broker to a single-instance broker, use the *static:* prefix and the OpenWire endpoint *uri* for *MyBroker2*. For example:

           ```xml
           <networkConnectors>
               <networkConnector name="connector_1_to_2" userName="myCommonUser" duplex="true"
               url="static:(ssl://b-12345b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-east-2.amazonaws.com:61617)/">
           </networkConnectors>
           ```

         - If you are connecting the broker to an active/standby broker, use the *masterslave:* prefix and the OpenWire endpoint *uri* for both brokers. For example:

           ```xml
           <networkConnectors>
               <networkConnector name="connector_1_to_2" userName="myCommonUser"
               duplex="true"
               url="masterslave:(ssl://b-12345b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-east-2.amazonaws.com:61617,ssl://b-9876l5k4-32ji-109h-8gfe-7d65c4b132al-2.mq.us-east-2.amazonaws.com:61617)/">
           </networkConnectors>
           ```

      - **Note**

         Don't include the password for the ActiveMQ user.

   f. Choose *Save*.
   g. In the *Save revision* dialog box, type *Add network of brokers connector for MyBroker2*.
   h. Choose *Save* to save the new revision of the configuration.

2. Edit *MyBroker1* to set the latest configuration revision to apply immediately.
a. On the MyBroker1 page, choose **Edit**.

b. On the **Edit MyBroker1** page, in the Configuration section, choose **Schedule Modifications**.

c. In the **Schedule broker modifications** section, choose to apply modifications **Immediately**.

d. Choose **Apply**.

**MyBroker1** is rebooted and your configuration revision is applied.

The network of brokers is created.

**Next Steps**

After you configure your network of brokers, you can test it by producing and consuming messages.

**Important**

Make sure that you enable inbound connections (p. 78) from your local machine for broker **MyBroker1** on port 8162 (for the ActiveMQ Web Console) and port 61617 (for the OpenWire endpoint).

You might also need to adjust your security group(s) settings to allow the producer and consumer to connect to the network of brokers.

1. On the **Amazon MQ console**, navigate to the **Connections** section and note the ActiveMQ Web Console endpoint for broker **MyBroker1**.

2. Navigate to the ActiveMQ Web Console for broker **MyBroker1**.

3. To verify that the network bridge is connected, choose **Network**.

   In the **Network Bridges** section, the name and the address of **MyBroker2** are listed in the Remote Broker and Remote Address columns.

4. From any machine that has access to broker **MyBroker2**, create a consumer. For example:

   ```
   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
   ```

   The consumer connects to the OpenWire endpoint of **MyBroker1** and begins to consume messages from queue **MyQueue**.

5. From any machine that has access to broker **MyBroker1**, create a producer and send some messages. For example:

   ```
   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
   ```

   The producer connects to the OpenWire endpoint of **MyBroker1** and begins to produce persistent messages to queue **MyQueue**.
Tutorial: Editing Broker Engine Version, Instance Type, CloudWatch Logs, and Maintenance Preferences

In addition to editing broker configurations and managing configuration revisions (p. 22), you can configure preferences specific to the broker.

**Note**
All preferences except for those for automatic minor version upgrades require you to schedule modifications. For more information, see Amazon MQ Broker Configuration Lifecycle (p. 60).

The following example shows how you can edit Amazon MQ broker preferences using the AWS Management Console.

**To Edit Broker Options**

1. Sign in to the Amazon MQ console.
2. From the broker list, select your broker (for example, *MyBroker*) and then choose *Edit*.
3. On the *Edit MyBroker* page, in the *Specifications* section, select a *Broker engine version* or a *Broker Instance type*.
4. In the *Configuration* section, select the configuration and revision for your broker. For more information, see Editing and Managing Broker Configurations (p. 22).
5. In the *Security and network* section, select a group from the *Security group(s)* drop-down, or choose *Create a new security group* to open the Amazon VPC console.
6. In the *CloudWatch Logs* section, choose whether to publish *General* logs and *Audit* logs to Amazon CloudWatch Logs. For more information, see Configuring Amazon MQ to Publish Logs to Amazon CloudWatch Logs (p. 110).

   **Important**
   If you don’t add the *CreateLogGroup* permission to your Amazon MQ user (p. 111) before the user creates or reboots the broker, Amazon MQ doesn’t create the log group.
   If you don’t configure a resource-based policy for Amazon MQ (p. 112), the broker can’t publish the logs to CloudWatch Logs.
7. In the *Maintenance* section, configure your broker’s maintenance schedule:
   To upgrade the broker to new versions as AWS releases them, choose *Enable automatic minor version upgrades*. Automatic upgrades occur during the *maintenance window* defined by the day of the week, the time of day (in 24-hour format), and the time zone (UTC by default).

   **Note**
   For an active/standby broker, if one of the broker instances undergoes maintenance, it takes Amazon MQ a short while to take the inactive instance out of service. This allows the healthy standby instance to become active and to begin accepting incoming communications.
8. Choose *Schedule modifications*.

   **Note**
   If you choose only *Enable automatic minor version upgrades*, the button changes to *Save* because no broker reboot is necessary.

Your preferences are applied to your broker at the specified time.
Tutorial: Creating and Applying Amazon MQ Broker Configurations

A configuration contains all of the settings for your ActiveMQ broker, in XML format (similar to ActiveMQ's activemq.xml file). You can create a configuration before creating any brokers. You can then apply the configuration to one or more brokers. You can apply a configuration immediately or during a maintenance window.

**Note**
For an active/standby broker, if one of the broker instances undergoes maintenance, it takes Amazon MQ a short while to take the inactive instance out of service. This allows the healthy standby instance to become active and to begin accepting incoming communications.

For more information, see the following:

- Configuration (p. 42)
- Amazon MQ Broker Configuration Lifecycle (p. 60)
- Amazon MQ Broker Configuration Parameters (p. 61)
- Editing and Managing Broker Configurations (p. 22)

The following example shows how you can create and apply an Amazon MQ broker configuration using the AWS Management Console.

**Topics**

- Step 1: Create a Configuration from Scratch (p. 20)
- Step 2: Create a New Configuration Revision (p. 21)
- Step 3: Apply a Configuration Revision to Your Broker (p. 21)

**Step 1: Create a Configuration from Scratch**

1. Sign in to the Amazon MQ console.
2. On the left, expand the navigation panel and choose **Configurations**.
3. On the **Configurations** page, choose **Create configuration**.
4. On the **Create configuration** page, in the Details section, type the **Configuration name** (for example, MyConfiguration) and select a **Broker engine** version.
   
   **Note**
   Currently, Amazon MQ supports only ActiveMQ broker engine versions 5.15.8, 5.15.6 and 5.15.0.
5. Choose **Create configuration**.
Step 2: Create a New Configuration Revision

1. From the configuration list, choose MyConfiguration.
   
   Note
   The first configuration revision is always created for you when Amazon MQ creates the configuration.

   On the MyConfiguration page, the broker engine type and version that your new configuration revision uses (for example, Apache ActiveMQ 5.15.8) are displayed.

2. On the Configuration details tab, the configuration revision number, description, and broker configuration in XML format are displayed.
   
   Note
   Editing the current configuration creates a new configuration revision.

3. Choose Edit configuration and make changes to the XML configuration.
4. Choose Save.
   
   The Save revision dialog box is displayed.
5. (Optional) Type A description of the changes in this revision.
6. Choose Save.
   
   The new revision of the configuration is saved.

   Important
   The Amazon MQ console automatically sanitizes invalid and prohibited configuration parameters according to a schema. For more information and a full list of permitted XML parameters, see Amazon MQ Broker Configuration Parameters (p. 61). Making changes to a configuration does not apply the changes to the broker immediately. To apply your changes, you must wait for the next maintenance window (p. 24) or reboot the broker (p. 32). For more information, see Amazon MQ Broker Configuration Lifecycle (p. 60).
   
   Currently, you can't delete a configuration.

Step 3: Apply a Configuration Revision to Your Broker

1. On the left, expand the navigation panel and choose Brokers.
2. From the broker list, select your broker (for example, MyBroker) and then choose Edit.
3. On the Edit MyBroker page, in the Configuration section, select a Configuration and a Revision and then choose Schedule Modifications.
4. In the Schedule broker modifications section, choose whether to apply modifications During the next scheduled maintenance window or Immediately.
   **Important**
   Your broker will be offline while it is being rebooted.
5. Choose Apply.
   Your configuration revision is applied to your broker at the specified time.

**Tutorial: Editing Amazon MQ Broker Configurations and Managing Configuration Revisions**

A configuration contains all of the settings for your ActiveMQ broker, in XML format (similar to ActiveMQ's activemq.xml file). You can apply a configuration immediately or during a maintenance window.

**Note**
For an active/standby broker, if one of the broker instances undergoes maintenance, it takes Amazon MQ a short while to take the inactive instance out of service. This allows the healthy standby instance to become active and to begin accepting incoming communications.

To keep track of the changes you make to your configuration, you can create configuration revisions.

For more information, see the following:

- Configuration (p. 42)
- Amazon MQ Broker Configuration Lifecycle (p. 60)
- Amazon MQ Broker Configuration Parameters (p. 61)
- Creating and Applying Broker Configurations (p. 20)

The following examples show how you can edit Amazon MQ broker configurations and manage broker configuration revisions using the AWS Management Console.

**Topics**

- To View a Previous Configuration Revision (p. 23)
- To Edit the Current Configuration Revision (p. 19)
- To Apply a Configuration Revision to Your Broker (p. 24)
To View a Previous Configuration Revision

1. Sign in to the Amazon MQ console.
2. From the broker list, select your broker (for example, MyBroker) and then choose Edit.
3. On the Edit MyBroker page, in the Configuration section, select a Configuration and a Revision and then choose Edit.

   Note
   Unless you select a configuration when you create a broker, the first configuration revision is always created for you when Amazon MQ creates the broker.

   On the MyBroker page, the broker engine type and version that the configuration uses (for example, Apache ActiveMQ 5.15.8) are displayed.

4. Choose Revision history.
5. The configuration Revision number, Revision date, and Description are displayed for each revision.
6. Select a revision and choose View details.

   The broker configuration in XML format is displayed.

To Edit the Current Configuration Revision

1. Sign in to the Amazon MQ console.
2. From the broker list, select your broker (for example, MyBroker) and then choose Edit.
4. On the Edit MyBroker page, in the Configuration section, select a Configuration and a Revision and then choose Edit.

   Note
   Unless you select a configuration when you create a broker, the first configuration revision is always created for you when Amazon MQ creates the broker.

   On the MyBroker page, the broker engine type and version that the configuration uses (for example, Apache ActiveMQ 5.15.8) are displayed.

5. On the Configuration details tab, the configuration revision number, description, and broker configuration in XML format are displayed.

   Note
   Editing the current configuration creates a new configuration revision.

6. Choose Edit configuration and make changes to the XML configuration.
To Apply a Configuration Revision to Your Broker

1. Sign in to the Amazon MQ console.
2. From the broker list, select your broker (for example, MyBroker) and then choose Edit.
3. On the Edit MyBroker page, in the Configuration section, select a Configuration and a Revision and then choose Schedule Modifications.
4. In the Schedule broker modifications section, choose whether to apply modifications During the next scheduled maintenance window or Immediately.
   Important
   Your broker will be offline while it is being rebooted.
5. Choose Apply.

   Your configuration revision is applied to your broker at the specified time.

To Roll Back Your Broker to the Last Configuration Revision

1. Sign in to the Amazon MQ console.
2. From the broker list, choose the name of your broker (for example, MyBroker).
3. On the MyBroker page, choose Actions, Roll back to last configuration.
4. (Optional) To review the Current configuration or the Last configuration, on the Roll back to the last configuration page, in the Summary section, choose Edit for either configuration.

5. In the Schedule broker modifications section, choose whether to apply modifications During the next scheduled maintenance window or Immediately.

   Important
   Your broker will be offline while it is being rebooted.

6. Choose Apply.

   Your configuration revision is applied to your broker at the specified time.

**Tutorial: Connecting a Java Application to Your Amazon MQ Broker**

After you create an Amazon MQ broker, you can connect your application to it. The following examples show how you can use the Java Message Service (JMS) to create a connection to the broker, create a queue, and send a message. For a complete, working Java example, see Working Java Example (p. 78).

You can connect to ActiveMQ brokers using various ActiveMQ clients. We recommend using the ActiveMQ Client.

**Topics**

- Prerequisites (p. 25)
- To Create a Message Producer and Send a Message (p. 26)
- To Create a Message Consumer and Receive the Message (p. 28)

**Prerequisites**

**Enable VPC Attributes**

To ensure that your broker is accessible within your VPC, you must enable the **enableDnsHostnames** and **enableDnsSupport** VPC attributes. For more information, see DNS Support in your VPC in the Amazon VPC User Guide.

**Enable Inbound Connections**

1. Sign in to the Amazon MQ console.
2. From the broker list, choose the name of your broker (for example, MyBroker).
3. On the MyBroker page, in the Connections section, note the addresses and ports of the broker’s ActiveMQ Web Console URL and wire-level protocols.
4. In the Details section, under Security and network, choose the name of your security group or Security Groups page of the EC2 Dashboard is displayed.
5. From the security group list, choose your security group.
6. At the bottom of the page, choose Inbound, and then choose Edit.
7. In the Edit inbound rules dialog box, add a rule for every URL or endpoint that you want to be publicly accessible (the following example shows how to do this for an ActiveMQ Web Console).
   a. Choose Add Rule.
   b. For Type, select Custom TCP.
   c. For Port Range, type the ActiveMQ Web Console port (8162).
   d. For Source, leave Custom selected and then type the IP address of the system that you want to be able to access the ActiveMQ Web Console (for example, 192.0.2.1).
   e. Choose Save.

Your broker can now accept inbound connections.

Add Java Dependencies

Add the activemq-client.jar and activemq-pool.jar packages to your Java class path. The following example shows these dependencies in a Maven project pom.xml file.

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

For more information about activemq-client.jar, see Initial Configuration in the Apache ActiveMQ documentation.

Important

In the following example code, producers and consumers run in a single thread. For production systems (or to test broker instance failover), make sure that your producers and consumers run on separate hosts or threads.

To Create a Message Producer and Send a Message

1. Create a JMS pooled connection factory for the message producer using your broker’s endpoint and then call the createConnection method against the factory.

   Note
   For an active/standby broker, Amazon MQ provides two ActiveMQ Web Console URLs, but only one URL is active at a time. Likewise, Amazon MQ provides two endpoints for each wire-level protocol, but only one endpoint is active in each pair at a time. The –1
and −2 suffixes denote a redundant pair. For more information, see Amazon MQ Broker Architecture (p. 45).

For wire-level protocol endpoints, you can allow your application to connect to either endpoint by using the Failover Transport.

```java
// Create a connection factory.
final ActiveMQConnectionFactory connectionFactory = new ActiveMQConnectionFactory(wireLevelEndpoint);

// Pass the username and password.
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();

Note
Message producers should always use the PooledConnectionFactory class. For more information, see Always Use Connection Pooling (p. 95).

2. Create a session, a queue named MyQueue, and a message producer.

```java
// 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. Create the message string "Hello from Amazon MQ!" and then send the message.

```java
// 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. Clean up the producer.

```java
producer.close();
producerSession.close();
producerConnection.close();
```
To Create a Message Consumer and Receive the Message

1. Create a JMS connection factory for the message producer using your broker's endpoint and then call the `createConnection` method against the factory.

```java
// Create a connection factory.
final ActiveMQConnectionFactory connectionFactory = new ActiveMQConnectionFactory(wireLevelEndpoint);

// Pass the username and password.
connectionFactory.setUserName(activeMqUsername);
connectionFactory.setPassword(activeMqPassword);

// Establish a connection for the consumer.
final Connection consumerConnection = connectionFactory.createConnection();
consumerConnection.start();
```

**Note**

Message consumers should *never* use the `PooledConnectionFactory` class. For more information, see Always Use Connection Pooling (p. 95).

2. Create a session, a queue named `MyQueue`, and a message consumer.

```java
// 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. Begin to wait for messages and receive the message when it arrives.

```java
// 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**

Unlike AWS messaging services (such as Amazon SQS), the consumer is constantly connected to the broker.

4. Close the consumer, session, and connection.

```java
consumer.close();
consumerSession.close();
consumerConnection.close();
```
Tutorial: Listing Amazon MQ Brokers and Viewing Broker Details

When you request that Amazon MQ create a broker, the creation process can take about 15 minutes.

The following example shows how you can confirm your broker's existence by listing your brokers in the current region using the AWS Management Console.

**To List Brokers and View Broker Details**

1. Sign in to the Amazon MQ console.
   
   Your brokers in the current region are listed.

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Deployment mode</th>
<th>Instance type</th>
</tr>
</thead>
<tbody>
<tr>
<td>MyBroker</td>
<td>Running</td>
<td>Single-instance broker</td>
<td>mq.m5.large</td>
</tr>
<tr>
<td>MyBroker2</td>
<td>Running</td>
<td>Active/standby broker for high availability</td>
<td>mq.m5.large</td>
</tr>
</tbody>
</table>

   The following information is displayed for each broker:
   
   - **Name**
   - **Creation date**
   - **Status (p. 42)**
   - **Deployment mode (p. 45)**
   - **Instance type (p. 39)**

2. Choose your broker's name (for example, **MyBroker**).

   On the **MyBroker** page, the **configured (p. 42) Details** are displayed for your broker:
Below the **Details** section, the following information is displayed:

- In the **Connections** section, the ActiveMQ Web Console URL and the wire-level protocol endpoints
- In the **Users** section, the users (p. 45) associated with the broker

### Tutorial: Creating and Managing Amazon MQ Broker Users

An ActiveMQ user is a person or an application that can access the queues and topics of an ActiveMQ broker. You can configure users to have specific permissions. For example, you can allow some users to access the ActiveMQ Web Console.

A group is a semantic label. You can assign a group to a user and configure permissions for groups to send to, receive from, and administer specific queues and topics.

**Note**

You can't configure groups independently of users. A group label is created when you add at least one user to it and deleted when you remove all users from it.
The following examples show how you can create, edit, and delete Amazon MQ broker users using the AWS Management Console.

Topics

- To Create a New User (p. 31)
- To edit an existing user (p. 31)
- To Delete an Existing User (p. 32)

To Create a New User

1. Sign in to the Amazon MQ console.
2. From the broker list, choose the name of your broker (for example, MyBroker) and then choose Edit.

On the MyBroker page, in the Users section, all the users for this broker are listed.

3. Choose Create user.
4. In the Create user dialog box, type a Username and Password.
5. (Optional) Type the names of groups to which the user belongs, separated by commas (for example: Devs, Admins).
6. (Optional) To enable the user to access the ActiveMQ Web Console, choose ActiveMQ Web Console.
7. Choose Create user.

Important
Making changes to a user does not apply the changes to the user immediately. To apply your changes, you must wait for the next maintenance window (p. 24) or reboot the broker (p. 32). For more information, see Amazon MQ Broker Configuration Lifecycle (p. 60).

To edit an existing user

1. Sign in to the Amazon MQ console.
2. From the broker list, choose the name of your broker (for example, MyBroker) and then choose Edit.

On the MyBroker page, in the Users section, all the users for this broker are listed.

3. Select a username and choose Edit.

The Edit user dialog box is displayed.
4. (Optional) Type a new Password.
5. (Optional) Add or remove the names of groups to which the user belongs, separated by commas (for example: Managers, Admins).
To Delete an Existing User

1. Sign in to the Amazon MQ console.
2. From the broker list, choose the name of your broker (for example, MyBroker) and then choose Edit.
3. On the MyBroker page, in the Users section, all the users for this broker are listed.

<table>
<thead>
<tr>
<th>Username</th>
<th>Console access</th>
<th>Groups</th>
</tr>
</thead>
<tbody>
<tr>
<td>paolo.santos</td>
<td>No</td>
<td>Devs</td>
</tr>
<tr>
<td>jane.doe</td>
<td>Yes</td>
<td>Admins</td>
</tr>
</tbody>
</table>

3. Select a username (for example, MyUser) and then choose Delete.
4. To confirm deleting the user, in the Delete MyUser? dialog box, choose Delete.

   Important  
   Making changes to a user does not apply the changes to the user immediately. To apply your changes, you must wait for the next maintenance window (p. 24) or reboot the broker (p. 32). For more information, see Amazon MQ Broker Configuration Lifecycle (p. 60).

Tutorial: Rebooting an Amazon MQ Broker

To apply a new configuration to a broker, you can reboot the broker. In addition, if your broker becomes unresponsive, you can reboot it to recover from a faulty state.

The following example shows how you can reboot an Amazon MQ broker using the AWS Management Console.

To Reboot an Amazon MQ Broker

1. Sign in to the Amazon MQ console.
2. From the broker list, choose the name of your broker (for example, MyBroker).

   Important  
   Your broker will be offline while it is being rebooted.
4. In the **Reboot broker** dialog box, choose **Reboot**.

Rebooting the broker takes about 5 minutes.

**Tutorial: Deleting an Amazon MQ Broker**

If you don’t use an Amazon MQ broker (and don’t foresee using it in the near future), it is a best practice to delete it from Amazon MQ to reduce your AWS costs.

The following example shows how you can delete a broker using the AWS Management Console.

**To Delete an Amazon MQ Broker**

1. Sign in to the Amazon MQ console.
2. From the broker list, select your broker (for example, **MyBroker**) and then choose **Delete**.
3. In the **Delete MyBroker?** dialog box, type **delete** and then choose **Delete**.

Deleting a broker takes about 5 minutes.

**Tutorial: Accessing CloudWatch Metrics for Amazon MQ**

Amazon MQ and Amazon CloudWatch are integrated so you can use CloudWatch to view and analyze metrics for your ActiveMQ broker and the broker’s destinations (queues and topics). You can view and analyze your Amazon MQ metrics from the CloudWatch console, the AWS CLI, or the CloudWatch CLI. CloudWatch metrics for Amazon MQ are automatically polled from the broker and then pushed to CloudWatch every minute.

For a full list of Amazon MQ metrics, see [Monitoring Amazon MQ Using CloudWatch](p. 103).

For information about creating a CloudWatch alarm for a metrics, see [Create or Edit a CloudWatch Alarm](in the Amazon CloudWatch User Guide).

**Note**

There is no charge for the Amazon MQ metrics reported in CloudWatch. These metrics are provided as part of the Amazon MQ service.

CloudWatch monitors only the first 200 destinations.
Topics

• AWS Management Console (p. 34)
• AWS Command Line Interface (p. 36)
• Amazon CloudWatch API (p. 36)

AWS Management Console

The following example shows you how to access CloudWatch metrics for Amazon MQ using the AWS Management Console.

**Note**

If you're already signed into the Amazon MQ console, on the broker **Details** page, choose **Actions, View CloudWatch metrics**.

1. Sign in to the CloudWatch console.
2. On the navigation panel, choose **Metrics**.
3. Select the **AmazonMQ** metric namespace.
4. Select one of the following metric dimensions:
   - Broker Metrics
   - Queue Metrics by Broker
   - Topic Metrics by Broker
In this example, **Broker Metrics** is selected.

5. You can now examine your Amazon MQ metrics:

- To sort the metrics, use the column heading.
- To graph the metric, select the check box next to the metric.
- To filter by metric, choose the metric name and then choose **Add to search**.
AWS Command Line Interface

To access Amazon MQ metrics using the AWS CLI, use the `get-metric-statistics` command.

For more information, see Get Statistics for a Metric in the Amazon CloudWatch User Guide.

Amazon CloudWatch API

To access Amazon MQ metrics using the CloudWatch API, use the `GetMetricStatistics` action.

For more information, see Get Statistics for a Metric in the Amazon CloudWatch User Guide.
How Amazon MQ Works

Amazon MQ makes it easy to create a message broker with the computing and storage resources that fit your needs. You can create, manage, and delete brokers using the AWS Management Console, Amazon MQ REST API, or the AWS Command Line Interface.

This section describes the basic elements of a message broker, lists available Amazon MQ broker instance types and their statuses, provides an overview of broker architecture, explains broker configuration parameters and offers a working example of using Java Message Service (JMS) with an ActiveMQ broker.

To learn about Amazon MQ REST APIs, see the Amazon MQ REST API Reference.

Topics
- Amazon MQ Basic Elements (p. 37)
- Amazon MQ Broker Architecture (p. 45)
- Amazon MQ Broker Configuration Parameters (p. 61)
- Working Examples of Using Java Message Service (JMS) with ActiveMQ (p. 78)
- Tagging resources (p. 85)

Amazon MQ Basic Elements

This section introduces key concepts essential to understanding Amazon MQ.

Topics
- Broker (p. 37)
- Configuration (p. 42)
- Engine (p. 43)
- Storage (p. 44)
- User (p. 45)

Broker

A broker is a message broker environment running on Amazon MQ. It is the basic building block of Amazon MQ. The combined description of the broker instance class (m5, t2) and size (large, micro) is a broker instance type (for example, mq.m5.large). For more information, see Broker Instance Types (p. 39).

- A single-instance broker is comprised of one broker in one Availability Zone. The broker communicates with your application and with Amazon EFS (by default) or with Amazon EBS.
- An active/standby broker is comprised of two brokers in two different Availability Zones, configured in a redundant pair. These brokers communicate synchronously with your application, and with Amazon EFS.

For more information, see Amazon MQ Broker Architecture (p. 45).

You can enable automatic minor version upgrades to new minor versions of the broker engine, as Apache releases new versions. Automatic upgrades occur during the maintenance window defined by the day of the week, the time of day (in 24-hour format), and the time zone (UTC by default).
For information about creating and managing brokers, see the following:

- Creating and Configuring a Broker (p. 11)
- Limits Related to Brokers (p. 100)
- Broker Statuses (p. 42)

**Supported Wire-level Protocols**

You can access your brokers by using any programming language that ActiveMQ supports and by enabling TLS explicitly for the following protocols:

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

**Attributes**

A broker has several attributes, for example:

- A name (MyBroker)
- An ID (b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9)
- An ActiveMQ Web Console URL (https://b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-east-2.amazonaws.com:8162)

For more information, see Web Console in the Apache ActiveMQ documentation.

**Important**

If you specify an authorization map which doesn't include the activemq-webconsole group, you can't use the ActiveMQ Web Console because the group isn't authorized to send messages to, or receive messages from, the Amazon MQ broker.

**Wire-level protocol endpoints:**

- 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**

This is an OpenWire endpoint.

- 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

For more information, see Configuring Transports in the Apache ActiveMQ documentation.
Note
For an active/standby broker, Amazon MQ provides two ActiveMQ Web Console URLs, but only one URL is active at a time. Likewise, Amazon MQ provides two endpoints for each wire-level protocol, but only one endpoint is active in each pair at a time. The -1 and -2 suffixes denote a redundant pair.

For a full list of broker attributes, see the following in the Amazon MQ REST API Reference:

- REST Operation ID: Broker
- REST Operation ID: Brokers
- REST Operation ID: Broker Reboot

Instance Types

The combined description of the broker instance class (m5, t2) and size (large, micro) is a broker instance type (for example, mq.m5.large). The following table lists the available Amazon MQ broker instance types.

Important
You can use Amazon EBS only with the mq.m5 broker instance type family. For more information, see Broker Storage (p. 44).

<table>
<thead>
<tr>
<th>Instance Type</th>
<th>vCPU</th>
<th>Memory (GiB)</th>
<th>Network Performance</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>mq.t2.micro</td>
<td>1</td>
<td>1</td>
<td>Low</td>
<td>Use the mq.t2.micro instance type for basic evaluation of Amazon MQ. This instance type (single-instance brokers only) qualifies for the AWS Free Tier. Use the mq.t2.micro instance type for basic evaluation of Amazon MQ. This instance type (single-instance brokers only) qualifies for the AWS Free Tier.</td>
</tr>
</tbody>
</table>

Note
Using the mq.t2.micro instance type is subject to CPU credits and baseline performance—with the ability to burst above the baseline level (for more information, see the AWS MQ Developer Guide).
<table>
<thead>
<tr>
<th>Instance Type</th>
<th>vCPU</th>
<th>Memory (GiB)</th>
<th>Network Performance</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>mq.m5.large</td>
<td>2</td>
<td>8</td>
<td>High</td>
<td>Use the mq.m5.large instance for regular development, testing, and production workloads.</td>
</tr>
</tbody>
</table>

For regular development, testing, and production workloads.

If your application requires fixed performance, consider using an mq.m5.large instance type.

CpuCreditBalance (p. 103 metric).
<table>
<thead>
<tr>
<th>Instance Type</th>
<th>vCPU</th>
<th>Memory (GiB)</th>
<th>Network Performance</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>mq.m5.xlarge</td>
<td>4</td>
<td>16</td>
<td>High</td>
<td>Use the mq.m5.xlarge, mq.m5.2xlarge, and mq.m5.4xlarge instance types for regular development, testing and production workloads that require high throughput.</td>
</tr>
<tr>
<td>mq.m5.2xlarge</td>
<td>8</td>
<td>32</td>
<td>High</td>
<td></td>
</tr>
<tr>
<td>mq.m5.4xlarge</td>
<td>16</td>
<td>64</td>
<td>High</td>
<td></td>
</tr>
</tbody>
</table>

**Note**
When your system uses persistent messages, its throughput depends on how quickly messages are consumed. If messages aren't consumed immediately, using larger instance types with persistent messages might not improve system throughput. In this case, we recommend setting the concurrentStoreAndDispatchQueues attribute to false. For more
Instance Type | vCPU | Memory (GiB) | Network Performance | Notes
--- | --- | --- | --- | ---
mq.m4.large | 2 | 8 | Moderate | Use the mq.m4.large instance type for compatibility with existing broker deployments. We recommend using an mq.m5.* instance for new brokers.

For more information about throughput considerations, see Choose the Correct Broker Instance Type for the Best Throughput (p. 97).

**Statuses**

A broker's current condition is indicated by a status. The following table lists the statuses of an Amazon MQ broker.

<table>
<thead>
<tr>
<th>Console</th>
<th>API</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creation failed</td>
<td>CREATION_FAILED</td>
<td>The broker couldn't be created.</td>
</tr>
<tr>
<td>Creation in progress</td>
<td>CREATION_IN_PROGRESS</td>
<td>The broker is currently being created.</td>
</tr>
<tr>
<td>Deletion in progress</td>
<td>DELETION_IN_PROGRESS</td>
<td>The broker is currently being deleted.</td>
</tr>
<tr>
<td>Reboot in progress</td>
<td>REBOOT_IN_PROGRESS</td>
<td>The broker is currently being rebooted.</td>
</tr>
<tr>
<td>Running</td>
<td>RUNNING</td>
<td>The broker is operational.</td>
</tr>
</tbody>
</table>

**Configuration**

A configuration contains all of the settings for your ActiveMQ broker, in XML format (similar to ActiveMQ's activemq.xml file). You can create a configuration before creating any brokers. You can then apply the configuration to one or more brokers.
Important
Making changes to a configuration does not apply the changes to the broker immediately. To apply your changes, you must wait for the next maintenance window (p. 24) or reboot the broker (p. 32). For more information, see Amazon MQ Broker Configuration Lifecycle (p. 60). Currently, you can't delete a configuration.

For information about creating, editing, and managing configurations, see the following:

- Creating and Applying Broker Configurations (p. 20)
- Editing and Managing Broker Configurations (p. 22)
- Limits Related to Configurations (p. 100)
- Amazon MQ Broker Configuration Parameters (p. 61)

To keep track of the changes you make to your configuration, you can create configuration revisions. For more information, see Creating and Applying Broker Configurations (p. 20) and Editing and Managing Broker Configurations (p. 22).

Attributes
A broker configuration has several attributes, for example:

- A name (MyConfiguration)
- An ID (c-1234a5b6-78cd-901e-2fgh-3i45j6k178l9)

For a full list of configuration attributes, see the following in the Amazon MQ REST API Reference:

- REST Operation ID: Configuration
- REST Operation ID: Configurations

For a full list of configuration revision attributes, see the following:

- REST Operation ID: Configuration Revision
- REST Operation ID: Configuration Revisions

Engine

A broker engine is a type of message broker that runs on Amazon MQ.

Amazon MQ supports the following versions of ActiveMQ:

- ActiveMQ 5.15.10 (recommended)
- ActiveMQ 5.15.9
- ActiveMQ 5.15.8
- ActiveMQ 5.15.6
- ActiveMQ 5.15.0
Storage

By default, Amazon MQ uses Amazon Elastic File System (Amazon EFS) for broker storage. To take advantage of high durability and replication across multiple Availability Zones, use Amazon EFS. To take advantage of low latency and high throughput, use Amazon EBS.

**Important**

- You can use Amazon EBS only with the `mq.m5` broker instance type family.
- Although you can change the *broker instance type*, you can’t change the *broker storage type* after you create the broker.
- Amazon EBS replicates data within a single Availability Zone and doesn’t support the *active/standby* (p. 49) deployment mode.
- When working with Amazon EBS, we recommend creating mechanisms that would allow your application to recreate the message data (if necessary), rather than using Amazon EBS as the sole message storage location for your broker. For example, you can use JMS or XA transactions or store your messages at a location from which they can be replayed or regenerated.

**Differences between Storage Types**

The following table provides a brief overview of the differences between in-memory, Amazon EFS, and Amazon EBS storage types.

<table>
<thead>
<tr>
<th>Storage Type</th>
<th>Persistence</th>
<th>Example Use Case</th>
<th>Approximate Maximum Number of Messages Enqueued per Producer, per Second (1KB Message)</th>
<th>Replication</th>
</tr>
</thead>
<tbody>
<tr>
<td>In-memory</td>
<td>Non-persistent</td>
<td>• Stock quotes&lt;br&gt;• Location data updates&lt;br&gt;• Frequently changed data</td>
<td>5,000</td>
<td>None</td>
</tr>
<tr>
<td>Amazon EBS</td>
<td>Persistent</td>
<td>• High volumes of text&lt;br&gt;• Order processing</td>
<td>500</td>
<td>Multiple copies within a single Availability Zone (AZ)</td>
</tr>
<tr>
<td>Amazon EFS</td>
<td>Persistent</td>
<td>Financial transactions</td>
<td>80</td>
<td>Multiple copies across multiple AZs</td>
</tr>
</tbody>
</table>

In-memory message storage provides the lowest latency and the highest throughput. However, messages are lost during instance replacement or broker restart.

Amazon EFS is designed to be highly durable, replicated across multiple AZs to prevent the loss of data resulting from the failure of any single component or an issue that affects the availability of an AZ. Amazon EBS is optimized for throughput and replicated across multiple servers within a single AZ.
**User**

An ActiveMQ user is a person or an application that can access the queues and topics of an ActiveMQ broker. You can configure users to have specific permissions. For example, you can allow some users to access the ActiveMQ Web Console.

A group is a semantic label. You can assign a group to a user and configure permissions for groups to send to, receive from, and administer specific queues and topics.

**Important**

Making changes to a user does *not* apply the changes to the user immediately. To apply your changes, you must wait for the next maintenance window (p. 24) or reboot the broker (p. 32). For more information, see Amazon MQ Broker Configuration Lifecycle (p. 60).

For information about users and groups, see the following in the Apache ActiveMQ documentation:

- Authorization
- Authorization Example

For information about creating, editing, and deleting ActiveMQ users, see the following:

- Creating and Managing Amazon MQ Broker Users (p. 30)
- Limits Related to Users (p. 101)

**Attributes**

For a full list of user attributes, see the following in the Amazon MQ REST API Reference:

- REST Operation ID: User
- REST Operation ID: Users

**Amazon MQ Broker Architecture**

Amazon MQ brokers can be created as *single-instance brokers* or *active/standby brokers*. For both deployment modes, Amazon MQ provides high durability by storing its data redundantly.

**Note**

Amazon MQ uses Apache KahaDB as its data store. Other data stores, such as JDBC and LevelDB, aren't supported.

You can access your brokers by using any programming language that ActiveMQ supports and by enabling TLS explicitly for the following protocols:

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

**Topics**
Amazon MQ Single-Instance Broker

A single-instance broker is comprised of one broker in one Availability Zone. The broker communicates with your application and with Amazon EFS (by default) or with Amazon EBS. For more information, see Broker Storage (p. 44).

The following diagram illustrates a single-instance broker with Amazon EFS storage replicated across multiple Availability Zones (AZs).
The following diagram illustrates a single-instance broker with Amazon EBS storage replicated across multiple servers within a single AZ.
Amazon MQ Active/Standby Broker for High Availability

An active/standby broker is comprised of two brokers in two different Availability Zones, configured in a redundant pair. These brokers communicate synchronously with your application, and with Amazon EFS. For more information, see Broker Storage (p. 44).

Usually, only one of the broker instances is active at any time, while the other broker instance is on standby. If one of the broker instances malfunctions or undergoes maintenance, it takes Amazon MQ a short while to take the inactive instance out of service. This allows the healthy standby instance to become active and to begin accepting incoming communications. When you reboot a broker, the failover takes only a few seconds.

For an active/standby broker, Amazon MQ provides two ActiveMQ Web Console URLs, but only one URL is active at a time. Likewise, Amazon MQ provides two endpoints for each wire-level protocol, but only one endpoint is active in each pair at a time. The –1 and –2 suffixes denote a redundant pair. For wire-level protocol endpoints, you can allow your application to connect to either endpoint by using the Failover Transport.

The following diagram illustrates an active/standby broker with Amazon EFS storage.
Amazon MQ Network of Brokers

Amazon MQ supports ActiveMQ's network of brokers feature.

A network of brokers is comprised of multiple simultaneously active single-instance brokers (p. 46) or active/standby brokers (p. 49). You can configure networks of brokers in a variety of topologies (p. 52) (for example, concentrator, hub-and-spokes, tree, or mesh), depending on your application's needs, such as high availability and scalability. For instance, a hub and spoke (p. 55) network of brokers can increase resiliency, preserving messages if one broker is not reachable. A network of brokers with a concentrator (p. 56) topology can collect messages from a larger number of brokers accepting incoming messages, and concentrate them to more central brokers, to better handle the load of many incoming messages.

For a tutorial and detailed configuration information, see the following:

- Creating and Configuring a Network of Brokers (p. 15)
- Configure Your Network of Brokers Correctly (p. 98)
• `networkConnector` (p. 75)
• `networkConnectionStartAsync` (p. 71)
• Networks of Brokers in the ActiveMQ documentation

The following are benefits of using a network of brokers:

• Creating a network of brokers allows you to increase your aggregate throughput and maximum producer and consumer connection count by adding broker instances.

• You can ensure better availability by allowing your producers and consumers to be aware of multiple active broker instances. This allows them to reconnect to a new instance if the one they're currently connected to becomes unavailable.

• Because producers and consumers can reconnect to another node in the network of brokers immediately, and because there's no need to wait for a standby broker instance to become promoted, client reconnection within a network of brokers is faster than for an active/standby broker for high availability (p. 49).

Topics

• How Does a Network of Brokers Work? (p. 51)
• How Does a Network of Brokers Handle Credentials? (p. 52)
• Sample Blueprints (p. 52)
• Network of Brokers Topologies (p. 52)
• Cross Region (p. 57)
• Dynamic Failover With Transport Connectors (p. 59)

How Does a Network of Brokers Work?

Amazon MQ supports the ActiveMQ network of brokers feature in a number of ways. First, you can edit the parameters within each broker's configuration to create a network of brokers, just as you would with native ActiveMQ. Second, Amazon MQ has sample blueprints that use AWS CloudFormation to automate the creation of a network of brokers. You can deploy these sample blueprints directly from the Amazon MQ console, or you can edit the related AWS CloudFormation templates to create your own topologies and configurations.

A network of brokers is established by connecting one broker to another using network connectors. Once connected, these brokers provide message forwarding. For instance, if `Broker1` establishes a network connector to `Broker2`, messages on `Broker1` are forwarded to `Broker2` if there is a consumer on that broker for the queue or topic. If the network connector is configured as `duplex`, messages are also forwarded from `Broker2` to `Broker1`. Network connectors are configured in the broker configuration.

See, Configuration (p. 42). For instance, here is an example `<networkConnector>` entry in a broker configuration:

```xml
<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>
```

A network of brokers ensures that messages flow from one broker instance to another, forwarding messages only to the broker instances that have corresponding consumers. For the benefit of broker instances adjacent to each other within the network, ActiveMQ sends messages to advisory topics about producers and consumers connecting to and disconnecting from the network. When a broker instance receives information about a producer that consumes from a particular destination, the broker instance begins to forward messages. For more information, see Advisory Topics in the ActiveMQ documentation.
How Does a Network of Brokers Handle Credentials?

For broker A to connect to broker B in a network, broker A must use valid credentials, like any other producer or consumer. Instead of providing a password in broker A's `<networkConnector>` configuration, you must first create a user on broker A with the same values as another user on broker B (these are separate, unique users that share the same username and password values). When you specify the `userName` attribute in the `<networkConnector>` configuration, Amazon MQ will add the password automatically at runtime.

**Important**

Don't specify the `password` attribute for the `<networkConnector>`. We don't recommend storing plaintext passwords in broker configuration files, because this makes the passwords visible in the Amazon MQ console. For more information, see Configure Network Connectors for Your Broker (p. 17).

Brokers must be in the same VPC or in peered VPCs. For more information, see Prerequisites (p. 16) in the Creating and Configuring a Network of Brokers (p. 15) tutorial.

Sample Blueprints

To get started using a Network of Brokers, Amazon MQ provides sample blueprints. These sample blueprints create a Network of Brokers deployment, and all related resources using, AWS CloudFormation. The two sample blueprints available are:

1. Mesh network of single instance brokers
2. Mesh network of active/standby brokers

From the Create brokers page, select one of the sample blueprints and choose Next. Once the resources have been created, review the generated brokers and their configurations in the Amazon MQ console.

By creating brokers and configuring different `networkConnector` elements in the broker configurations, you can create a network of brokers in many different topologies. For more information on configuring a network of brokers, see Networks of Brokers in the ActiveMQ documentation.

Network of Brokers Topologies

By deploying brokers, and then configuring `networkConnector` entries in their configurations, you can build a network of brokers using different network topologies. A network connector provides on-
demand message forwarding between connected brokers. Connections can be configured as duplex, where messages are forwarded both ways between brokers, or not duplex, where the forwarding only propagates from one broker to the other. For example, if we have a duplex connection between Broker1 and Broker2, messages will be forwarded from each to the other if there is a consumer.

![Diagram of two brokers connected by a duplex network connector](image1)

With a duplex network connector, messages are forwarded from each broker to the other. These are forwarded on-demand: if there is a consumer on Broker2 for a message on Broker1, the message is forwarded. Similarly, if there is a consumer on Broker1 for a message on Broker2 the message is also forwarded.

For non-duplex connections, messages are forwarded only from one broker to the other. In this example, if there is a consumer on Broker2 for a message on Broker1, the message is forwarded. But messages will not be forwarded from Broker2 to Broker1.

![Diagram of two brokers connected by a non-duplex network connector](image2)

Using both duplex and non-duplex network connectors, it is possible to build a network of brokers in any number of network topologies.

**Note**

In each of the network topology examples, the `networkConnector` elements reference the endpoint of the brokers they connect to. Replace the broker endpoint entries in the `uri` attributes with the endpoints of your brokers. See, Listing Brokers and Viewing Broker Details (p. 29).

**Mesh Topology**

A mesh topology provides multiple brokers that are all connected to each other. This simple example connects three single-instance brokers, but you can configure more brokers as a mesh.
This topology, and one that includes a mesh of active/standby pairs of brokers, can be created using sample blueprints in the Amazon MQ console. You can create these sample blueprint deployment to see a working network of brokers, and review how they are configured.

You can configure a three broker mesh network like this by adding a network connector to Broker1 that makes duplex connections to both Broker2 and Broker3, and a single duplex connection between Broker2 and Broker3.

**Network connectors for Broker1:**

```xml
<networkConnectors>
  <networkConnector name="connector_1_to_2" userName="myCommonUser" duplex="true"
    uri="static:(ssl://b-987615k4-32ji-109h-8gfe-7d65c4b132a1-2.mq.us-east-2.amazonaws.com:61617)="/>
  <networkConnector name="connector_1_to_3" userName="myCommonUser" duplex="true"
    uri="static:(ssl://b-743c885d-2244-4c95-af67-a85017ff234e-3.mq.us-east-2.amazonaws.com:61617)="/>
</networkConnectors>
```

**Network connectors for Broker2:**

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

By adding the above connectors to the configurations of Broker1 and Broker2, you can create a mesh between these three brokers that forwards message between all the brokers on demand. For more information, see Amazon MQ Broker Configuration Parameters (p. 61).
Hub and Spoke Topology

In a hub and spoke topology, messages are preserved if there is a disruption to any broker on a spoke. Messages are forwarded throughout, and only the central Broker1 is critical to the network’s operation.

To configure the hub and spoke network of brokers in this example, you could add a networkConnector to each of the brokers on the spokes in the configuration of Broker1.

```xml
<networkConnectors>
  <networkConnector name="connector_hub_and_spoke_2" userName="myCommonUser" duplex="true"
    uri="static:(ssl://b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-east-2.amazonaws.com:61617)"/>
  <networkConnector name="connector_hub_and_spoke_3" userName="myCommonUser" duplex="true"
    uri="static:(ssl://b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-east-2.amazonaws.com:61617)"/>
</networkConnectors>
```
Concentrator Topology

In this example topology, the three brokers on the bottom can handle a large number of connections, and those messages are concentrated to Broker1 and Broker2. Each of the other brokers has a non-duplex connection to the more central brokers. To scale the capacity of this topology, you can add additional brokers that receive messages and concentrate those messages in Broker1 and Broker2.

To configure this topology, each of the brokers on the bottom would contain a network connector to each of the brokers they are concentrating messages to.

*Network connectors for Broker3:*

```xml
<networkConnectors>
  <networkConnector name="3_to_1" userName="myCommonUser" duplex="false"
                     uri="static:(ssl://b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-east-2.amazonaws.com:61617)"/>
  <networkConnector name="3_to_2" userName="myCommonUser" duplex="false"
                     uri="static:(ssl://b-9876l5k4-32ji-109h-8gfe-7d65c4b132a1-2.mq.us-east-2.amazonaws.com:61617)"/>
</networkConnectors>
```
Network connectors for Broker4:

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

Network connectors for Broker5:

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

Cross Region

To configure a network of brokers that spans AWS regions, deploy brokers in those regions, and configure network connectors to the endpoints of those brokers.
To configure a network of brokers like this example, you could add `networkConnectors` entries to the configurations of `Broker1` and `Broker4` that reference the wire-level endpoints of those brokers.

**Network connectors for Broker1:**

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

Network connector for Broker2:

<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>

Network connectors for Broker4:

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

Dynamic Failover With Transport Connectors

In addition to configuring networkConnector elements, you can configure your broker transportConnector options to enable dynamic failover, and to rebalance connections when brokers are added or removed from the network.

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

In this example both updateClusterClients and rebalanceClusterClients are set to true. In this case clients will be provided a list of brokers in the network, and will request them to rebalance if a new broker joins.

Available options:

• updateClusterClients: Passes information to clients about changes in the network of broker topology.
• rebalanceClusterClients: Causes clients to re-balance across brokers when a new broker is added to a network of brokers.
• updateClusterClientsOnRemove: Updates clients with topology information when a broker leaves a network of brokers.

When updateClusterClients is set to true, clients can be configured to connect to a single broker in a network of brokers.
When a new broker connects, it will be receive a list of URIs of all brokers in the network. If the connection to the broker fails, it can dynamically switch to one of the brokers provided when it connected.

For more information on failover, see Broker-side Options for Failover in the Active MQ documentation.

**Amazon MQ Broker Configuration Lifecycle**

Making changes to a configuration revision or an ActiveMQ user does *not* apply the changes immediately. To apply your changes, you must wait for the next maintenance window (p. 24) or reboot the broker (p. 32). For more information, see Amazon MQ Broker Configuration Lifecycle (p. 60).

The following diagram illustrates the configuration lifecycle.

**Important**

The next scheduled maintenance window triggers a reboot. If the broker is rebooted before the next scheduled maintenance window, the changes are applied after the reboot.

For information about creating, editing, and managing configurations, see the following:

- Creating and Applying Broker Configurations (p. 20)
For information about creating, editing, and deleting ActiveMQ users, see the following:

- Creating and Managing Amazon MQ Broker Users (p. 30)
- Limits Related to Users (p. 101)

**Amazon MQ Broker Configuration Parameters**

A *configuration* contains all of the settings for your ActiveMQ broker, in XML format (similar to ActiveMQ's `activemq.xml` file). You can create a configuration before creating any brokers. You can then apply the configuration to one or more brokers. For more information, see the following:

- Configuration (p. 42)
- Creating and Applying Broker Configurations (p. 20)
- Editing and Managing Broker Configurations (p. 22)
- Limits Related to Configurations (p. 100)

**Working with Spring XML Configuration Files**

ActiveMQ brokers are configured using Spring XML files. You can configure many aspects of your ActiveMQ broker, such as predefined destinations, destination policies, authorization policies, and plugins. Amazon MQ controls some of these configuration elements, such as network transports and storage. Other configuration options, such as creating networks of brokers, aren't currently supported.

The full set of supported configuration options is specified in the Amazon MQ XML schemas. Download zip files of the supported schemas using the following links.

- `amazon-mq-active-mq-5.15.10.xsd.zip`
- `amazon-mq-active-mq-5.15.9.xsd.zip`
- `amazon-mq-active-mq-5.15.8.xsd.zip`
- `amazon-mq-active-mq-5.15.6.xsd.zip`
- `amazon-mq-active-mq-5.15.0.xsd.zip`

You can use these schemas to validate and sanitize your configuration files. Amazon MQ also lets you provide configurations by uploading XML files. When you upload an XML file, Amazon MQ automatically sanitizes and removes invalid and prohibited configuration parameters according to the schema.

**Note**

You can use only static values for attributes. Amazon MQ sanitizes elements and attributes that contain Spring expressions, variables, and element references from your configuration.

**Topics**

- Elements Permitted in Amazon MQ Configurations (p. 62)
- Elements and Their Attributes Permitted in Amazon MQ Configurations (p. 64)
- Elements, Child Collection Elements, and Their Child Elements Permitted in Amazon MQ Configurations (p. 71)
Elements Permitted in Amazon MQ Configurations

The following is a detailed listing of the elements permitted in Amazon MQ configurations. For more information, see XML Configuration in the Apache ActiveMQ documentation.

<table>
<thead>
<tr>
<th>Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>abortSlowAckConsumerStrategy</td>
</tr>
<tr>
<td>abortSlowConsumerStrategy</td>
</tr>
<tr>
<td>authorizationEntry</td>
</tr>
<tr>
<td>authorizationMap</td>
</tr>
<tr>
<td>authorizationPlugin</td>
</tr>
<tr>
<td>broker</td>
</tr>
<tr>
<td>cachedMessageGroupMapFactory</td>
</tr>
<tr>
<td>compositeQueue</td>
</tr>
<tr>
<td>compositeTopic</td>
</tr>
<tr>
<td>constantPendingMessageLimitStrategy</td>
</tr>
<tr>
<td>discarding</td>
</tr>
<tr>
<td>discardingDLQBrokerPlugin</td>
</tr>
<tr>
<td>fileCursor</td>
</tr>
<tr>
<td>fileDurableSubscriberCursor</td>
</tr>
<tr>
<td>fileQueueCursor</td>
</tr>
<tr>
<td>filteredDestination</td>
</tr>
<tr>
<td>fixedCountSubscriptionRecoveryPolicy</td>
</tr>
<tr>
<td>fixedSizedSubscriptionRecoveryPolicy</td>
</tr>
<tr>
<td>forcePersistencyModeBrokerPlugin</td>
</tr>
<tr>
<td>individualDeadLetterStrategy</td>
</tr>
<tr>
<td>lastImageSubscriptionRecoveryPolicy</td>
</tr>
<tr>
<td>messageGroupHashBucketFactory</td>
</tr>
<tr>
<td>mirroredQueue</td>
</tr>
<tr>
<td>noSubscriptionRecoveryPolicy</td>
</tr>
<tr>
<td>oldestMessageEvictionStrategy</td>
</tr>
<tr>
<td>oldestMessageWithLowestPriorityEvictionStrategy</td>
</tr>
<tr>
<td>policyEntry</td>
</tr>
<tr>
<td>policyMap</td>
</tr>
</tbody>
</table>

62
<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>prefetchRatePendingMessageLimitStrategy</td>
<td>(attributes) (p. 68)</td>
<td></td>
</tr>
<tr>
<td>priorityDispatchPolicy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>priorityNetworkDispatchPolicy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>queryBasedSubscriptionRecoveryPolicy</td>
<td>(attributes) (p. 69)</td>
<td></td>
</tr>
<tr>
<td>queue</td>
<td>(attributes) (p. 69)</td>
<td></td>
</tr>
<tr>
<td>redeliveryPlugin</td>
<td>(attributes (p. 69)</td>
<td>child collection elements) (p. 73)</td>
</tr>
<tr>
<td>redeliveryPolicy</td>
<td>(attributes) (p. 69)</td>
<td></td>
</tr>
<tr>
<td>redeliveryPolicyMap</td>
<td>(child collection elements) (p. 74)</td>
<td></td>
</tr>
<tr>
<td>retainedMessageSubscriptionRecoveryPolicy</td>
<td>(child collection elements) (p. 74)</td>
<td></td>
</tr>
<tr>
<td>roundRobinDispatchPolicy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sharedDeadLetterStrategy</td>
<td>(attributes (p. 69)</td>
<td>child collection elements) (p. 74)</td>
</tr>
<tr>
<td>simpleDispatchPolicy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>simpleMessageGroupMapFactory</td>
<td></td>
<td></td>
</tr>
<tr>
<td>statisticsBrokerPlugin</td>
<td></td>
<td></td>
</tr>
<tr>
<td>storeCursor</td>
<td></td>
<td></td>
</tr>
<tr>
<td>storeDurableSubscriberCursor</td>
<td>(attributes) (p. 69)</td>
<td></td>
</tr>
<tr>
<td>strictOrderDispatchPolicy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tempDestinationAuthorizationEntry</td>
<td>(attributes) (p. 69)</td>
<td></td>
</tr>
<tr>
<td>tempQueue</td>
<td>(attributes) (p. 70)</td>
<td></td>
</tr>
<tr>
<td>tempTopic</td>
<td>(attributes) (p. 70)</td>
<td></td>
</tr>
<tr>
<td>timedSubscriptionRecoveryPolicy</td>
<td>(attributes) (p. 70)</td>
<td></td>
</tr>
<tr>
<td>timeStamppingBrokerPlugin</td>
<td>(attributes) (p. 70)</td>
<td></td>
</tr>
<tr>
<td>topic</td>
<td>(attributes) (p. 70)</td>
<td></td>
</tr>
<tr>
<td>transportConnector</td>
<td>(attributes) (p. 70)</td>
<td></td>
</tr>
<tr>
<td>uniquePropertyMessageEvictionStrategy</td>
<td>(attributes) (p. 70)</td>
<td></td>
</tr>
<tr>
<td>virtualDestinationInterceptor</td>
<td>(child collection elements) (p. 74)</td>
<td></td>
</tr>
<tr>
<td>virtualTopic</td>
<td>(attributes) (p. 70)</td>
<td></td>
</tr>
<tr>
<td>vmCursor</td>
<td></td>
<td></td>
</tr>
<tr>
<td>vmDurableCursor</td>
<td></td>
<td></td>
</tr>
<tr>
<td>vmQueueCursor</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


Elements and Their Attributes Permitted in Amazon MQ Configurations

The following is a detailed listing of the elements and their attributes permitted in Amazon MQ configurations. For more information, see XML Configuration in the Apache ActiveMQ documentation.

<table>
<thead>
<tr>
<th>Element</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>abortSlowAckConsumerStrategy</td>
<td>abortConnection</td>
</tr>
<tr>
<td></td>
<td>checkPeriod</td>
</tr>
<tr>
<td></td>
<td>ignoreIdleConsumers</td>
</tr>
<tr>
<td></td>
<td>ignoreNetworkConsumers</td>
</tr>
<tr>
<td></td>
<td>maxSlowCount</td>
</tr>
<tr>
<td></td>
<td>maxSlowDuration</td>
</tr>
<tr>
<td></td>
<td>maxTimeSinceLastAck</td>
</tr>
<tr>
<td></td>
<td>name</td>
</tr>
<tr>
<td>abortSlowConsumerStrategy</td>
<td>abortConnection</td>
</tr>
<tr>
<td></td>
<td>checkPeriod</td>
</tr>
<tr>
<td></td>
<td>ignoreNetworkConsumers</td>
</tr>
<tr>
<td></td>
<td>maxSlowCount</td>
</tr>
<tr>
<td></td>
<td>maxSlowDuration</td>
</tr>
<tr>
<td></td>
<td>name</td>
</tr>
<tr>
<td>authorizationEntry</td>
<td>admin</td>
</tr>
<tr>
<td></td>
<td>queue</td>
</tr>
<tr>
<td></td>
<td>read</td>
</tr>
<tr>
<td></td>
<td>tempQueue</td>
</tr>
<tr>
<td></td>
<td>tempTopic</td>
</tr>
<tr>
<td></td>
<td>topic</td>
</tr>
<tr>
<td></td>
<td>write</td>
</tr>
<tr>
<td>broker</td>
<td>advisorySupport</td>
</tr>
<tr>
<td></td>
<td>allowTempAutoCreationOnSend</td>
</tr>
<tr>
<td></td>
<td>cacheTempDestinations</td>
</tr>
<tr>
<td></td>
<td>consumerSystemUsagePortion</td>
</tr>
<tr>
<td></td>
<td>dedicatedTaskRunner</td>
</tr>
<tr>
<td></td>
<td>deleteAllMessagesOnStartup</td>
</tr>
<tr>
<td>Element</td>
<td>Attribute</td>
</tr>
<tr>
<td>--------------------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>keepDurableSubsActive</td>
</tr>
<tr>
<td></td>
<td>maxPurgedDestinationsPerSweep</td>
</tr>
<tr>
<td></td>
<td>monitorConnectionSplits</td>
</tr>
<tr>
<td></td>
<td><strong>networkConnectorStartAsync</strong> (p. 71)</td>
</tr>
<tr>
<td></td>
<td>offlineDurableSubscriberTaskSchedule</td>
</tr>
<tr>
<td></td>
<td>offlineDurableSubscriberTimeout</td>
</tr>
<tr>
<td></td>
<td>persistenceThreadPriority</td>
</tr>
<tr>
<td></td>
<td>persistent</td>
</tr>
<tr>
<td></td>
<td>populateJMSXUserID</td>
</tr>
<tr>
<td></td>
<td>producerSystemUsagePortion</td>
</tr>
<tr>
<td></td>
<td>rejectDurableConsumers</td>
</tr>
<tr>
<td></td>
<td>rollbackOnlyOnAsyncException</td>
</tr>
<tr>
<td></td>
<td>schedulePeriodForDestinationPurge</td>
</tr>
<tr>
<td></td>
<td>schedulerSupport</td>
</tr>
<tr>
<td></td>
<td>splitSystemUsageForProducersConsumers</td>
</tr>
<tr>
<td></td>
<td>taskRunnerPriority</td>
</tr>
<tr>
<td></td>
<td>timeBeforePurgeTempDestinations</td>
</tr>
<tr>
<td></td>
<td>useAuthenticatedPrincipalForJMSXUserID</td>
</tr>
<tr>
<td></td>
<td>useMirroredQueues</td>
</tr>
<tr>
<td></td>
<td>useTempMirroredQueues</td>
</tr>
<tr>
<td></td>
<td>useVirtualDestSubs</td>
</tr>
<tr>
<td></td>
<td>useVirtualDestSubsOnCreation</td>
</tr>
<tr>
<td></td>
<td>useVirtualTopics</td>
</tr>
<tr>
<td>cachedMessageGroupMapFactory</td>
<td>cacheSize</td>
</tr>
<tr>
<td>compositeQueue</td>
<td>concurrentSend</td>
</tr>
<tr>
<td></td>
<td>copyMessage</td>
</tr>
<tr>
<td></td>
<td>forwardOnly</td>
</tr>
<tr>
<td></td>
<td>name</td>
</tr>
<tr>
<td>compositeTopic</td>
<td>concurrentSend</td>
</tr>
<tr>
<td></td>
<td>copyMessage</td>
</tr>
<tr>
<td></td>
<td>forwardOnly</td>
</tr>
</tbody>
</table>

65
<table>
<thead>
<tr>
<th>Element</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>constantPendingMessageLimitStrategy</td>
<td>limit</td>
</tr>
<tr>
<td>discarding</td>
<td>deadLetterQueue</td>
</tr>
<tr>
<td></td>
<td>enableAudit</td>
</tr>
<tr>
<td></td>
<td>expiration</td>
</tr>
<tr>
<td></td>
<td>maxAuditDepth</td>
</tr>
<tr>
<td></td>
<td>maxProducersToAudit</td>
</tr>
<tr>
<td></td>
<td>processExpired</td>
</tr>
<tr>
<td></td>
<td>processNonPersistent</td>
</tr>
<tr>
<td>discardingDLQBrokerPlugin</td>
<td>dropAll</td>
</tr>
<tr>
<td></td>
<td>dropOnly</td>
</tr>
<tr>
<td></td>
<td>dropTemporaryQueues</td>
</tr>
<tr>
<td></td>
<td>dropTemporaryTopics</td>
</tr>
<tr>
<td></td>
<td>reportInterval</td>
</tr>
<tr>
<td>filteredDestination</td>
<td>queue</td>
</tr>
<tr>
<td></td>
<td>selector</td>
</tr>
<tr>
<td></td>
<td>topic</td>
</tr>
<tr>
<td>fixedCountSubscriptionRecoveryPolicy</td>
<td>maximumSize</td>
</tr>
<tr>
<td>fixedSizedSubscriptionRecoveryPolicy</td>
<td>maximumSize</td>
</tr>
<tr>
<td></td>
<td>useSharedBuffer</td>
</tr>
<tr>
<td>forcePersistencyModeBrokerPlugin</td>
<td>persistenceFlag</td>
</tr>
<tr>
<td>individualDeadLetterStrategy</td>
<td>destinationPerDurableSubscriber</td>
</tr>
<tr>
<td></td>
<td>enableAudit</td>
</tr>
<tr>
<td></td>
<td>expiration</td>
</tr>
<tr>
<td></td>
<td>maxAuditDepth</td>
</tr>
<tr>
<td></td>
<td>maxProducersToAudit</td>
</tr>
<tr>
<td></td>
<td>processExpired</td>
</tr>
<tr>
<td></td>
<td>processNonPersistent</td>
</tr>
<tr>
<td></td>
<td>queuePrefix</td>
</tr>
<tr>
<td></td>
<td>queueSuffix</td>
</tr>
<tr>
<td></td>
<td>topicPrefix</td>
</tr>
</tbody>
</table>
## Permitted Attributes

<table>
<thead>
<tr>
<th>Element</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>topicSuffix</td>
<td>useQueueForQueueMessages</td>
</tr>
<tr>
<td>useQueueForTopicMessages</td>
<td>bucketCount</td>
</tr>
<tr>
<td>cacheSize</td>
<td>copyMessage</td>
</tr>
<tr>
<td>postfix</td>
<td>prefix</td>
</tr>
<tr>
<td>mirroredQueue</td>
<td>evictExpiredMessagesHighWatermark</td>
</tr>
<tr>
<td>oldestMessageEvictionStrategy</td>
<td>oldestMessageWithLowestPriorityEvictionStrategy</td>
</tr>
<tr>
<td>policyEntry</td>
<td>advisoryForConsumed</td>
</tr>
<tr>
<td>advisoryForDelivery</td>
<td>advisoryForDiscardingMessages</td>
</tr>
<tr>
<td>advisoryForFastProducers</td>
<td>advisoryForSlowConsumers</td>
</tr>
<tr>
<td>advisoryWhenFull</td>
<td>allConsumersExclusiveByDefault</td>
</tr>
<tr>
<td>alwaysRetroactive</td>
<td>blockedProducerWarningInterval</td>
</tr>
<tr>
<td>consumersBeforeDispatchStarts</td>
<td>cursorMemoryHighWaterMark</td>
</tr>
<tr>
<td>gcOptimizeMessageStorage</td>
<td>durableTopicPrefetch</td>
</tr>
<tr>
<td>enableAudit</td>
<td>expireMessagesPeriod</td>
</tr>
<tr>
<td>gcInactiveDestinations</td>
<td>gcWithNetworkConsumers</td>
</tr>
<tr>
<td>inactiveTimeoutBeforeGC</td>
<td>inactiveTimeoutBeforeGC</td>
</tr>
<tr>
<td>includeBodyForAdvisory</td>
<td>lazyDispatch</td>
</tr>
<tr>
<td>Element</td>
<td>Attribute</td>
</tr>
<tr>
<td>---------------</td>
<td>----------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>maxAuditDepth</td>
</tr>
<tr>
<td></td>
<td>maxBrowsePageSize</td>
</tr>
<tr>
<td></td>
<td>maxDestinations</td>
</tr>
<tr>
<td></td>
<td>maxExpirePageSize</td>
</tr>
<tr>
<td></td>
<td>maxPageSize</td>
</tr>
<tr>
<td></td>
<td>maxProducersToAudit</td>
</tr>
<tr>
<td></td>
<td>maxQueueAuditDepth</td>
</tr>
<tr>
<td></td>
<td>memoryLimit</td>
</tr>
<tr>
<td></td>
<td>messageGroupMapFactoryType</td>
</tr>
<tr>
<td></td>
<td>minimumMessageSize</td>
</tr>
<tr>
<td></td>
<td>optimizedDispatch</td>
</tr>
<tr>
<td></td>
<td>optimizeMessageStoreInFlightLimit</td>
</tr>
<tr>
<td></td>
<td>persistJMSRedelivered</td>
</tr>
<tr>
<td></td>
<td>prioritizedMessages</td>
</tr>
<tr>
<td></td>
<td>producerFlowControl</td>
</tr>
<tr>
<td></td>
<td>queue</td>
</tr>
<tr>
<td></td>
<td>queueBrowserPrefetch</td>
</tr>
<tr>
<td></td>
<td>queuePrefetch</td>
</tr>
<tr>
<td></td>
<td>reduceMemoryFootprint</td>
</tr>
<tr>
<td></td>
<td>sendAdvisoryIfNoConsumers</td>
</tr>
<tr>
<td></td>
<td>storeUsageHighWaterMark</td>
</tr>
<tr>
<td></td>
<td>strictOrderDispatch</td>
</tr>
<tr>
<td></td>
<td>tempQueue</td>
</tr>
<tr>
<td></td>
<td>tempTopic</td>
</tr>
<tr>
<td></td>
<td>timeBeforeDispatchStarts</td>
</tr>
<tr>
<td></td>
<td>topic</td>
</tr>
<tr>
<td></td>
<td>topicPrefetch</td>
</tr>
<tr>
<td></td>
<td>useCache</td>
</tr>
<tr>
<td></td>
<td>useConsumerPriority</td>
</tr>
<tr>
<td></td>
<td>usePrefetchExtension</td>
</tr>
<tr>
<td></td>
<td>prefetchRatePendingMessageLimitStrategy multiplier</td>
</tr>
<tr>
<td></td>
<td>multiplier</td>
</tr>
<tr>
<td><strong>Element</strong></td>
<td><strong>Attribute</strong></td>
</tr>
<tr>
<td>----------------------------------------------------------------</td>
<td>---------------------------------------------------</td>
</tr>
<tr>
<td><code>queryBasedSubscriptionRecoveryPolicy</code></td>
<td><code>query</code></td>
</tr>
<tr>
<td><code>queue</code></td>
<td><code>DLQ</code></td>
</tr>
<tr>
<td></td>
<td><code>physicalName</code></td>
</tr>
<tr>
<td><code>redeliveryPlugin</code></td>
<td><code>fallbackToDeadLetter</code></td>
</tr>
<tr>
<td></td>
<td><code>sendToDlqIfMaxRetriesExceeded</code></td>
</tr>
<tr>
<td><code>redeliveryPolicy</code></td>
<td><code>backOffMultiplier</code></td>
</tr>
<tr>
<td></td>
<td><code>collisionAvoidancePercent</code></td>
</tr>
<tr>
<td></td>
<td><code>initialRedeliveryDelay</code></td>
</tr>
<tr>
<td></td>
<td><code>maximumRedeliveries</code></td>
</tr>
<tr>
<td></td>
<td><code>maximumRedeliveryDelay</code></td>
</tr>
<tr>
<td></td>
<td><code>preDispatchCheck</code></td>
</tr>
<tr>
<td></td>
<td><code>queue</code></td>
</tr>
<tr>
<td></td>
<td><code>redeliveryDelay</code></td>
</tr>
<tr>
<td></td>
<td><code>tempQueue</code></td>
</tr>
<tr>
<td></td>
<td><code>tempTopic</code></td>
</tr>
<tr>
<td></td>
<td><code>topic</code></td>
</tr>
<tr>
<td></td>
<td><code>useCollisionAvoidance</code></td>
</tr>
<tr>
<td></td>
<td><code>useExponentialBackOff</code></td>
</tr>
<tr>
<td><code>sharedDeadLetterStrategy</code></td>
<td><code>enableAudit</code></td>
</tr>
<tr>
<td></td>
<td><code>expiration</code></td>
</tr>
<tr>
<td></td>
<td><code>maxAuditDepth</code></td>
</tr>
<tr>
<td></td>
<td><code>maxProducersToAudit</code></td>
</tr>
<tr>
<td></td>
<td><code>processExpired</code></td>
</tr>
<tr>
<td></td>
<td><code>processNonPersistent</code></td>
</tr>
<tr>
<td><code>storeDurableSubscriberCursor</code></td>
<td><code>immediatePriorityDispatch</code></td>
</tr>
<tr>
<td></td>
<td><code>useCache</code></td>
</tr>
<tr>
<td><code>tempDestinationAuthorizationEntry</code></td>
<td><code>admin</code></td>
</tr>
<tr>
<td></td>
<td><code>queue</code></td>
</tr>
<tr>
<td></td>
<td><code>read</code></td>
</tr>
<tr>
<td></td>
<td><code>tempQueue</code></td>
</tr>
<tr>
<td></td>
<td><code>tempTopic</code></td>
</tr>
<tr>
<td>Element</td>
<td>Attribute</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>topic</td>
<td>topic</td>
</tr>
<tr>
<td>write</td>
<td>write</td>
</tr>
<tr>
<td>tempQueue</td>
<td>DLQ</td>
</tr>
<tr>
<td>physicalName</td>
<td></td>
</tr>
<tr>
<td>tempTopic</td>
<td>DLQ</td>
</tr>
<tr>
<td>physicalName</td>
<td></td>
</tr>
<tr>
<td>timedSubscriptionRecoveryPolicy</td>
<td>zeroExpirationOverride</td>
</tr>
<tr>
<td>timeStampingBrokerPlugin</td>
<td>recoverDuration</td>
</tr>
<tr>
<td>futureOnly</td>
<td></td>
</tr>
<tr>
<td>processNetworkMessages</td>
<td></td>
</tr>
<tr>
<td>ttlCeiling</td>
<td></td>
</tr>
<tr>
<td>topic</td>
<td>DLQ</td>
</tr>
<tr>
<td>physicalName</td>
<td></td>
</tr>
<tr>
<td>transportConnector</td>
<td>•</td>
</tr>
<tr>
<td>name</td>
<td></td>
</tr>
<tr>
<td>updateClusterClients</td>
<td></td>
</tr>
<tr>
<td>rebalanceClusterClients</td>
<td></td>
</tr>
<tr>
<td>updateClusterClientsOnRemove</td>
<td></td>
</tr>
<tr>
<td>uniquePropertyMessageEvictionStrategy</td>
<td>evictExpiredMessagesHighWatermark</td>
</tr>
<tr>
<td>propertyName</td>
<td></td>
</tr>
<tr>
<td>virtualTopic</td>
<td>concurrentSend</td>
</tr>
<tr>
<td>local</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td></td>
</tr>
<tr>
<td>postfix</td>
<td></td>
</tr>
<tr>
<td>prefix</td>
<td></td>
</tr>
<tr>
<td>selectorAware</td>
<td></td>
</tr>
<tr>
<td>transactedSend</td>
<td></td>
</tr>
</tbody>
</table>

**Amazon MQ Parent Element Attributes**

The following is a detailed explanation of parent element attributes. For more information, see [XML Configuration](#) in the Apache ActiveMQ documentation.
Topics

- broker (p. 71)

broker

broker is a parent collection element.

Attributes

networkConnectionStartAsync

To mitigate network latency and to allow other networks to start in a timely manner, use the `<networkConnectionStartAsync>` tag. The tag instructs the broker to use an executor to start network connections in parallel, asynchronous to a broker start.

Default: false

Example Configuration

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

Elements, Child Collection Elements, and Their Child Elements Permitted in Amazon MQ Configurations

The following is a detailed listing of the elements, child collection elements, and their child elements permitted in Amazon MQ configurations. For more information, see XML Configuration in the Apache ActiveMQ documentation.

<table>
<thead>
<tr>
<th>Element</th>
<th>Child Collection Element</th>
<th>Child Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>authorizationMap</td>
<td>authorizationEntries</td>
<td>authorizationEntry (p. 74)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tempDestinationAuthorizationEntry</td>
</tr>
<tr>
<td></td>
<td>defaultEntry</td>
<td>authorizationEntry</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tempDestinationAuthorizationEntry</td>
</tr>
<tr>
<td></td>
<td>tempDestinationAuthorizationEntry</td>
<td></td>
</tr>
<tr>
<td>authorizationPlugin</td>
<td>map</td>
<td>authorizationMap</td>
</tr>
<tr>
<td>broker</td>
<td>destinationInterceptors</td>
<td>mirroredQueue</td>
</tr>
<tr>
<td></td>
<td></td>
<td>virtualDestinationInterceptor</td>
</tr>
<tr>
<td></td>
<td>destinationPolicy</td>
<td>policyMap</td>
</tr>
<tr>
<td></td>
<td>destinations</td>
<td>queue</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tempQueue</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tempTopic</td>
</tr>
<tr>
<td></td>
<td>networkConnectors</td>
<td>networkConnector (p. 75)</td>
</tr>
<tr>
<td>Element</td>
<td>Child Collection Element</td>
<td>Child Element</td>
</tr>
<tr>
<td>----------------------</td>
<td>--------------------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>persistenceAdapter</td>
<td></td>
<td>kahaDB (p. 76)</td>
</tr>
<tr>
<td>plugins</td>
<td>authorizationPlugin</td>
<td></td>
</tr>
<tr>
<td></td>
<td>discardingDLQBrokerPlugin</td>
<td></td>
</tr>
<tr>
<td></td>
<td>forcePersistencyModeBrokerPlugin</td>
<td></td>
</tr>
<tr>
<td></td>
<td>redeliveryPlugin</td>
<td></td>
</tr>
<tr>
<td></td>
<td>statisticsBrokerPlugin</td>
<td></td>
</tr>
<tr>
<td></td>
<td>timeStampingBrokerPlugin</td>
<td></td>
</tr>
<tr>
<td>systemUsage</td>
<td>systemUsage (p. 77)</td>
<td></td>
</tr>
<tr>
<td>transportConnector</td>
<td>name</td>
<td></td>
</tr>
<tr>
<td></td>
<td>updateClusterClients</td>
<td></td>
</tr>
<tr>
<td></td>
<td>rebalanceClusterClients</td>
<td></td>
</tr>
<tr>
<td></td>
<td>updateClusterClientsOnRemove</td>
<td></td>
</tr>
<tr>
<td>compositeQueue</td>
<td>forwardTo</td>
<td>queue</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tempQueue</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tempTopic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>topic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>filteredDestination</td>
</tr>
<tr>
<td>compositeTopic</td>
<td>forwardTo</td>
<td>queue</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tempQueue</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tempTopic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>topic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>filteredDestination</td>
</tr>
<tr>
<td>policyEntry</td>
<td>deadLetterStrategy</td>
<td>discarding</td>
</tr>
<tr>
<td></td>
<td></td>
<td>individualDeadLetterStrategy</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sharedDeadLetterStrategy</td>
</tr>
<tr>
<td></td>
<td>destination</td>
<td>queue</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tempQueue</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tempTopic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>topic</td>
</tr>
<tr>
<td></td>
<td>dispatchPolicy</td>
<td>priorityDispatchPolicy</td>
</tr>
<tr>
<td></td>
<td></td>
<td>priorityNetworkDispatchPolicy</td>
</tr>
<tr>
<td>Element</td>
<td>Child Collection Element</td>
<td>Child Element</td>
</tr>
<tr>
<td>---------</td>
<td>--------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td></td>
<td>roundRobinDispatchPolicy</td>
<td></td>
</tr>
<tr>
<td></td>
<td>simpleDispatchPolicy</td>
<td></td>
</tr>
<tr>
<td></td>
<td>strictOrderDispatchPolicy</td>
<td></td>
</tr>
<tr>
<td></td>
<td>messageEvictionStrategy</td>
<td>oldestMessageEvictionStrategy</td>
</tr>
<tr>
<td></td>
<td></td>
<td>oldestMessageWithLowestPriorityEvictionStrategy</td>
</tr>
<tr>
<td></td>
<td></td>
<td>uniquePropertyMessageEvictionStrategy</td>
</tr>
<tr>
<td></td>
<td>messageGroupMapFactory</td>
<td>cachedMessageGroupMapFactory</td>
</tr>
<tr>
<td></td>
<td></td>
<td>messageGroupHashBucketFactory</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simpleMessageGroupMapFactory</td>
</tr>
<tr>
<td></td>
<td>pendingDurableSubscriberPolicy</td>
<td>fileDurableSubscriberCursor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>storeDurableSubscriberCursor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>vmDurableCursor</td>
</tr>
<tr>
<td></td>
<td>pendingMessageLimitStrategy</td>
<td>constantPendingMessageLimitStrategy</td>
</tr>
<tr>
<td></td>
<td></td>
<td>prefetchRatePendingMessageLimitStrategy</td>
</tr>
<tr>
<td></td>
<td>pendingQueuePolicy</td>
<td>fileQueueCursor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>storeCursor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>vmQueueCursor</td>
</tr>
<tr>
<td></td>
<td>pendingSubscriberPolicy</td>
<td>fileCursor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>vmCursor</td>
</tr>
<tr>
<td></td>
<td>slowConsumerStrategy</td>
<td>abortSlowAckConsumerStrategy</td>
</tr>
<tr>
<td></td>
<td></td>
<td>abortSlowConsumerStrategy</td>
</tr>
<tr>
<td></td>
<td>subscriptionRecoveryPolicy</td>
<td>fixedCountSubscriptionRecoveryPolicy</td>
</tr>
<tr>
<td></td>
<td></td>
<td>fixedSizedSubscriptionRecoveryPolicy</td>
</tr>
<tr>
<td></td>
<td></td>
<td>lastImageSubscriptionRecoveryPolicy</td>
</tr>
<tr>
<td></td>
<td></td>
<td>noSubscriptionRecoveryPolicy</td>
</tr>
<tr>
<td></td>
<td></td>
<td>queryBasedSubscriptionRecoveryPolicy</td>
</tr>
<tr>
<td></td>
<td></td>
<td>retainedMessageSubscriptionRecoveryPolicy</td>
</tr>
<tr>
<td></td>
<td></td>
<td>timedSubscriptionRecoveryPolicy</td>
</tr>
<tr>
<td></td>
<td>policyMap</td>
<td>defaultEntry</td>
</tr>
<tr>
<td></td>
<td></td>
<td>policyEntry</td>
</tr>
<tr>
<td></td>
<td>policyEntries</td>
<td>policyEntry</td>
</tr>
<tr>
<td></td>
<td>redeliveryPlugin</td>
<td>redeliveryPolicyMap</td>
</tr>
<tr>
<td></td>
<td></td>
<td>redeliveryPolicyMap</td>
</tr>
</tbody>
</table>
### Amazon MQ Child Element Attributes

The following is a detailed explanation of child element attributes. For more information, see [XML Configuration](https://activemq.apache.org/) in the Apache ActiveMQ documentation.

**Topics**
- authorizationEntry (p. 74)
- networkConnector (p. 75)
- kahaDB (p. 76)
- systemUsage (p. 77)

**authorizationEntry**

authorizationEntry is a child of the authorizationEntries child collection element.

**Attributes**

`admin|read|write`

The permissions granted to a group of users. For more information, see [Always Configure an Authorization Map](https://activemq.apache.org/) (p. 94).

**Default:** null
Example Configuration

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

**networkConnector**

*networkConnector* is a child of the *networkConnectors* child collection element.

**Topics**
- Attributes (p. 75)
- Example Configurations (p. 76)

**Attributes**

**conduitSubscriptions**

Specifies whether a network connection in a network of brokers treats multiple consumers subscribed to the same destination as one consumer. For example, if *conduitSubscriptions* is set to *true* and two consumers connect to broker B and consume from a destination, broker B combines the subscriptions into a single logical subscription over the network connection to broker A, so that only a single copy of a message is forwarded from broker A to broker B.

**Note**

Setting *conduitSubscriptions* to *true* can reduce redundant network traffic. However, using this attribute can have implications for the load-balancing of messages across consumers and might cause incorrect behavior in certain scenarios (for example, with JMS message selectors or with durable topics).

**Default:** *true*

**duplex**

Specifies whether the connection in the network of brokers is used to produce and consume messages. For example, if broker A creates a connection to broker B in non-duplex mode, messages can be forwarded only from broker A to broker B. However, if broker A creates a duplex connection to broker B, then broker B can forward messages to broker A without having to configure a *<networkConnector>*.

**Default:** *false*

**name**

The name of the bridge in the network of brokers.

**Default:** *bridge*

**uri**

The wire-level protocol endpoint for one of two brokers (or for multiple brokers) in a network of brokers.
**Default:** null

**username**

The username common to the brokers in a network of brokers.

**Default:** null

### Example Configurations

**Note**

When using a `networkConnector` to define a network of brokers, don't include the password for the user common to your brokers.

#### A Network of Brokers with Two Brokers

In this configuration, two brokers are connected in a network of brokers. The name of the network connector is `connector_1_to_2`, the username common to the brokers is `myCommonUser`, the connection is duplex, and the OpenWire endpoint URI is prefixed by `static:`, indicating a one-to-one connection between the brokers.

```xml
<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>
```

For more information, see Configure Network Connectors for Your Broker (p. 17).

#### A Network of Brokers with Multiple Brokers

In this configuration, multiple brokers are connected in a network of brokers. The name of the network connector is `connector_1_to_2`, the username common to the brokers is `myCommonUser`, the connection is duplex, and the comma-separated list of OpenWire endpoint URIs is prefixed by `masterslave:`, indicating a failover connection between the brokers. The failover from broker to broker isn't randomized and reconnection attempts continue indefinitely.

```xml
<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-7d65c4b132al-2.mq.us-east-2.amazonaws.com:61617)"/>
</networkConnectors>
```

**Note**

We recommend using the `masterslave:` prefix for networks of brokers. The prefix is identical to the more explicit `static:failover:()?randomize=false&maxReconnectAttempts=0` syntax.

**kahaDB**

kahaDB is a child of the `persistenceAdapter` child collection element.

**Attributes**

**concurrentStoreAndDispatchQueues**

Specifies whether to use concurrent store and dispatch for queues. For more information, see Disable Concurrent Store and Dispatch for Queues with Slow Consumers (p. 97).

**Default:** true
journalDiskSyncInterval

Interval (ms) for when to perform a disk sync if journalDiskSyncStrategy=periodic. For more information, see the Apache ActiveMQ kahaDB documentation.

**Default:** 1000

journalDiskSyncStrategy

From ActiveMQ 5.14.0: this setting configures the disk sync policy. For more information, see the Apache ActiveMQ kahaDB documentation.

**Default:** always

**Note**

The ActiveMQ documentation states that the data loss is limited to the duration of journalDiskSyncInterval, which has a default of 1s. The data loss can be longer than the interval, but it is difficult to be precise. Use caution.

preallocationStrategy

configures how the broker will try to preallocate the journal files when a new journal file is needed. For more information, see the Apache ActiveMQ kahaDB documentation.

**Default:** sparse_file

Example Configuration

**Example**

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

systemUsage

systemUsage is a child of the systemUsage child collection element. It controls the maximum amount of space the broker will use before slowing down producers. For more information, see Producer Flow Control in the Apache ActiveMQ documentation.

**Child Element**

memoryUsage

memoryUsage is a child of the systemUsage child element. It manages memory usage. Use memoryUsage to keep track of how much of something is being used so that you can control working set usage productively. For more information, see the schema in the Apache ActiveMQ documentation.

**Child Element**

memoryUsage is a child of the memoryUsage child element.

**Attribute**

percentOfJvmHeap

Integer between 0 (inclusive) and 70 (inclusive).

**Default:** 70
Attributes

sendFailIfNoSpace

Sets whether a `send()` method should fail if there is no space free. The default value is false, which blocks the `send()` method until space becomes available. For more information, see the schema in the Apache ActiveMQ documentation.

**Default:** false

sendFailIfNoSpaceAfterTimeout

**Default:** null

Example Configuration

Example

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

Working Examples of Using Java Message Service (JMS) with ActiveMQ

The following examples show how you can work with ActiveMQ programmatically:

- The OpenWire example Java code connects to a broker, creates a queue, and sends and receives a message. For a detailed breakdown and explanation, see Connecting a Java Application to Your Broker (p. 25).
- The MQTT example Java code connects to a broker, creates a topic, and publishes and receives a message.
- The STOMP+WSS example Java code connects to a broker, creates a queue, and publishes and receives a message.

Prerequisites

Enable VPC Attributes

To ensure that your broker is accessible within your VPC, you must enable the `enableDnsHostnames` and `enableDnsSupport` VPC attributes. For more information, see DNS Support in your VPC in the Amazon VPC User Guide.

Enable Inbound Connections

1. Sign in to the Amazon MQ console.
2. From the broker list, choose the name of your broker (for example, MyBroker).
3. On the **MyBroker** page, in the **Connections** section, note the addresses and ports of the broker’s ActiveMQ Web Console URL and wire-level protocols.

4. In the **Details** section, under **Security and network**, choose the name of your security group or [Security Groups](#). The **Security Groups** page of the EC2 Dashboard is displayed.

5. From the security group list, choose your security group.

6. At the bottom of the page, choose **Inbound**, and then choose **Edit**.

7. In the **Edit inbound rules** dialog box, add a rule for every URL or endpoint that you want to be publicly accessible (the following example shows how to do this for an ActiveMQ Web Console).
   
   a. Choose **Add Rule**.
   
   b. For **Type**, select **Custom TCP**.
   
   c. For **Port Range**, type the ActiveMQ Web Console port (8162).
   
   d. For **Source**, leave **Custom** selected and then type the IP address of the system that you want to be able to access the ActiveMQ Web Console (for example, 192.0.2.1).
   
   e. Choose **Save**.

   Your broker can now accept inbound connections.

### Add Java Dependencies

#### OpenWire

Add the `activemq-client.jar` and `activemq-pool.jar` packages to your Java class path. The following example shows these dependencies in a Maven project `pom.xml` file.

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

For more information about `activemq-client.jar`, see Initial Configuration in the Apache ActiveMQ documentation.

#### MQTT

Add the `org.eclipse.paho.client.mqttv3.jar` package to your Java class path. The following example shows this dependency in a Maven project `pom.xml` file.

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

For more information about `org.eclipse.paho.client.mqttv3.jar`, see Eclipse Paho Java Client.
Add the following packages to your Java class path:

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

The following example shows these dependencies in a Maven project pom.xml file.

```xml
<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>
```

For more information, see [STOMP Support](https://docs.spring.io/spring-framework/docs/current/javadoc-api) in the Spring Framework documentation.

**Important**

In the following example code, producers and consumers run in a single thread. For production systems (or to test broker instance failover), make sure that your producers and consumers run on separate hosts or threads.

```java
 OFFSET
 /*
 */
```
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);
        final Destination producerDestination = producerSession.createQueue("MyQueue");
        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);
    }

    private static void receiveMessage(Connection connection) throws JMSException {
        // Create a consumer.
        final Session consumerSession = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
        final Queue consumerQueue = consumerSession.createQueue("MyQueue");
        final MessageConsumer consumer = consumerSession.createConsumer(consumerQueue);

        // Consume messages from the queue.
        final Message consumerMessage = consumer.receive();
        System.out.println("Received message: " + consumerMessage.getText());
    }
}

81
// 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 username and password.
    connectionFactory.setUserName(ACTIVE_MQ_USERNAME);
    connectionFactory.setPassword(ACTIVE_MQ_PASSWORD);
    return connectionFactory;
}
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();
        connOpts.setUserName(ACTIVE_MQ_USERNAME);
        connOpts.setPassword(ACTIVE_MQ_PASSWORD.toCharArray());
        client.connect(connOpts);
        client.setCallback(this);
        client.subscribe("+");

        // Create a message.
        final MqttMessage message = new MqttMessage(text.getBytes());
        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();
    }
}
```java
@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

import org.springframework.messaging.converter.StringMessageConverter;
import org.springframework.messaging.simp.stomp.*;
import org.springframework.web.socket.WebSocketHttpHeaders;
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-123456abcdefg-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);
    }
}

Tagging resources

Amazon MQ supports resource tagging to help track your cost allocation. You can tag resources when creating them, or by viewing the details of that resource.

Topics
- Tagging for Cost Allocation (p. 86)
- Managing Tags in the Amazon MQ Console (p. 86)
- Managing Using Amazon MQ API Actions (p. 87)
Tagging for Cost Allocation

To organize and identify your Amazon MQ resources for cost allocation, you can add metadata tags that identify the purpose of a broker or configuration. This is especially useful when you have many brokers. You can use cost allocation tags to organize your AWS bill to reflect your own cost structure. To do this, sign up to get your AWS account bill to include the tag keys and values. For more information, see Setting Up a Monthly Cost Allocation Report in the AWS Billing and Cost Management User Guide.

For instance, you could add tags that represent the cost center and purpose of your Amazon MQ resources:

<table>
<thead>
<tr>
<th>Resource</th>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Broker1</td>
<td>Cost Center</td>
<td>34567</td>
</tr>
<tr>
<td></td>
<td>Stack</td>
<td>Production</td>
</tr>
<tr>
<td>Broker2</td>
<td>Cost Center</td>
<td>34567</td>
</tr>
<tr>
<td></td>
<td>Stack</td>
<td>Production</td>
</tr>
<tr>
<td>Broker3</td>
<td>Cost Center</td>
<td>12345</td>
</tr>
<tr>
<td></td>
<td>Stack</td>
<td>Development</td>
</tr>
</tbody>
</table>

This tagging scheme allows you to group two state machines performing related tasks in the same cost center, while tagging an unrelated broker with a different cost allocation tag.

Managing Tags in the Amazon MQ Console

Adding Tags to New Resources

Amazon MQ lets you to add tags to resources as they are created. You can quickly add tags to the resources you are creating in the Amazon MQ console.

To add tags as you create a new broker:

1. From the Create a broker page, select Additional settings.
2. Under Tags, select Add tag.
3. Enter a Key and Value pair.
4. (Optional) Select Add tag to add multiple tags to your broker.
5. Select Create broker.

To add tags as you create a configuration:
1. From the **Create configuration** page, select **Advanced**.
2. Under **Tags** on the **Create configuration** page, select **Add tag**.
3. Enter a **Key** and **Value** pair.
4. (Optional) Select **Add tag** to add multiple tags to your configuration.
5. Select **Create configuration**.

### Viewing and Managing Tags for Existing Resources

Amazon MQ allows you to view and manage the tags for your resources in the Amazon MQ console. You can manage tags for an individual resource by editing the tags on the details page for that resource. To edit tags on Amazon MQ resources:

1. Select either **Brokers** or **Configurations** in the Amazon MQ console.
   
   Under the **Tags** section, review the existing tags for that resource.
2. To add new or manage existing tags, select **Edit** (or **Create tag** if have no existing tags).
3. Update tags for your resource:
   - To modify existing tags, edit the **Key** and **Value**.
   - To remove existing tags, select **Remove**.
   - To add a new tag, select **Add tag** and enter a **Key** and **Value**.
4. Select **Save**.

### Managing Using Amazon MQ API Actions

Amazon MQ allows you to view and manage the tags of your resources using the REST API.

For more information, see the [Amazon MQ REST API Reference](#).
Migrating to Amazon MQ

Use the following topics to get started with migrating your on-premises message broker to Amazon MQ.

Topics
- Migrating to Amazon MQ without Service Interruption (p. 88)
- Migrating to Amazon MQ with Service Interruption (p. 90)

For detailed information and examples, see Migrating from RabbitMQ to Amazon MQ in the AWS Compute Blog.

Migrating to Amazon MQ without Service Interruption

The following diagrams illustrate the scenario of migrating from an on-premises message broker to an Amazon MQ broker in the AWS Cloud without service interruption.

Important
This scenario might cause messages to be delivered out of order. If you're concerned about message ordering, follow the steps in Migrating to Amazon MQ with Service Interruption (p. 90).
To migrate to Amazon MQ without service interruption

1. Create and configure an Amazon MQ broker (p. 11) and note your broker’s endpoint, for example:

   \[ \text{ssl://b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-east-2.amazonaws.com:61617} \]

2. For either of the following cases, use the Failover Transport to allow your consumers to randomly connect to your on-premises broker’s endpoint or your Amazon MQ broker’s endpoint. For example:

   \[ \text{failover:(ssl://on-premises-broker.example.com:61617,ssl://b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.mq.us-east-2.amazonaws.com:61617)?randomize=true} \]

Do one of the following:

- One by one, point each existing consumer to your Amazon MQ broker’s endpoint.
• Create new consumers and point them to your Amazon MQ broker's endpoint.

  **Note**  
  If you scale up your consumer fleet during the migration process, it is a best practice to scale it down afterward.

1. One by one, stop each existing producer, point the producer to your Amazon MQ broker's endpoint, and then restart the producer.

2. Wait for your consumers to drain the destinations on your on-premises broker.

3. Change your consumers' Failover transport to include only your Amazon MQ broker's endpoint. For example:

   ```
   failover:(ssl://b-1234a5b6-78cd-901e-2fgh-3i45j6k17819-1.mq.us-east-2.amazonaws.com:61617)
   ```

4. Stop your on-premises broker.

---

**Migrating to Amazon MQ with Service Interruption**

The following diagrams illustrate the scenario of migrating from an on-premises message broker to an Amazon MQ broker in the AWS Cloud with service interruption.

**Important**  
This scenario requires you to point your producer to your Amazon MQ broker's endpoint before you do the same for your consumers. This sequence ensures that any messages in a FIFO (first-in-first-out) queue maintain their order during the migration process. If you're not concerned about message ordering, follow the steps in Migrating to Amazon MQ without Service Interruption (p. 88).
To migrate to Amazon MQ with service interruption

1. Create and configure an Amazon MQ broker (p. 11) and note your broker's endpoint, for example:

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

2. Stop your existing producer, point the producer to your Amazon MQ broker's endpoint, and then restart the producer.

   **Important**
   This step requires an interruption of your application's functionality because no consumers are yet consuming messages from the Amazon MQ broker.

3. Wait for your consumers to drain the destinations on your on-premises broker.

4. Do one of the following:
   - One by one, point each existing consumer to your Amazon MQ broker's endpoint.
   - Create new consumers and point them to your Amazon MQ broker's endpoint.
To migrate to Amazon MQ with service interruption

Note
If you scale up your consumer fleet during the migration process, it is a best practice to scale it down afterward.

Stop your on-premises broker.
Best Practices for Amazon MQ

Use these best practices to make the most of Amazon MQ.

Topics
- Using Amazon MQ Securely (p. 93)
- Connecting to Amazon MQ (p. 94)
- Ensuring Effective Amazon MQ Performance (p. 96)
- Avoid Slow Restarts by Recovering Prepared XA Transactions (p. 98)

Using Amazon MQ Securely

The following design patterns can improve the security of your Amazon MQ broker.

Topics
- Prefer Brokers without Public Accessibility (p. 93)
- Always Use Client-Side Encryption as a Complement to TLS (p. 93)
- Always Configure an Authorization Map (p. 94)
- Block Unnecessary Protocols with VPC Security Groups (p. 94)

Prefer Brokers without Public Accessibility

Brokers created without public accessibility can’t be accessed from outside of your VPC. This greatly reduces your broker’s susceptibility to Distributed Denial of Service (DDoS) attacks from the public internet. For more information, see Accessing the ActiveMQ Web Console of a Broker without Public Accessibility (p. 14) in this guide and How to Help Prepare for DDoS Attacks by Reducing Your Attack Surface on the AWS Security Blog.

Always Use Client-Side Encryption as a Complement to TLS

You can access your brokers using the following protocols with TLS enabled:

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

Amazon MQ encrypts messages at rest and in transit using encryption keys that it manages and stores securely. For additional security, we highly recommend designing your application to use client-side encryption. For more information, see the AWS Encryption SDK Developer Guide.
Always Configure an Authorization Map

Because ActiveMQ has no authorization map configured by default, any authenticated user can perform any action on the broker. Thus, it is a best practice to restrict permissions by group. For more information, see authorizationEntry (p. 74).

Block Unnecessary Protocols with VPC Security Groups

To improve security, you should restrict the connections of unnecessary protocols and ports by properly configuring your Amazon VPC Security Group. For instance, to restrict access to most protocols while allowing access to OpenWire and the ActiveMQ web console, you could allow access to only 61617 and 8162. This limits your exposure by blocking protocols you are not using, while allowing OpenWire and the ActiveMQ web console to function normally.

Allow only the protocol ports that you are using.

- AMQP: 5671
- MQTT: 8883
- OpenWire: 61617
- STOMP: 61614
- WebSocket: 61619

For more information see.

- Configure Additional Broker Settings (p. 12)
- Security Groups for your VPC
- Default Security Group for Your VPC
- Working with Security Groups

Connecting to Amazon MQ

The following design patterns can improve the effectiveness of your application's connection to your Amazon MQ broker.

Topics

- Never Modify or Delete the Amazon MQ Elastic Network Interface (p. 94)
- Always Use Connection Pooling (p. 95)
- Always Use the Failover Transport to Connect to Multiple Broker Endpoints (p. 96)
- Avoid Using Message Selectors (p. 96)
- Prefer Virtual Destinations to Durable Subscriptions (p. 96)

Never Modify or Delete the Amazon MQ Elastic Network Interface

When you first create an Amazon MQ broker (p. 11), Amazon MQ provisions an elastic network interface in the Virtual Private Cloud (VPC) under your account and, thus, requires a number of EC2 permissions (p. 117). The network interface allows your client (producer or consumer) to communicate
Always Use Connection Pooling

In a scenario with a single producer and single consumer (such as the Getting Started with Amazon MQ (p. 5) tutorial), you can use a single `ActiveMQConnectionFactory` class for every producer and consumer. For example:

```java
// Create a connection factory.
final ActiveMQConnectionFactory connectionFactory = new ActiveMQConnectionFactory(wireLevelEndpoint);

// Pass the username and password.
connectionFactory.setUserName(activeMqUsername);
connectionFactory.setPassword(activeMqPassword);

// Establish a connection for the consumer.
final Connection consumerConnection = connectionFactory.createConnection();
consumerConnection.start();
```

However, in more realistic scenarios with multiple producers and consumers, it can be costly and inefficient to create a large number of connections for multiple producers. In these scenarios, you should group multiple producer requests using the `PooledConnectionFactory` class. For example:

```java
// Create a connection factory.
final PooledConnectionFactory pooledConnectionFactory = new PooledConnectionFactory(wireLevelEndpoint);

// Pass the username and password.
pooledConnectionFactory.setUserName(activeMqUsername);
pooledConnectionFactory.setPassword(activeMqPassword);

// Establish a connection for the consumer.
final Connection consumerConnection = pooledConnectionFactory.createConnection();
consumerConnection.start();
```

Note
---
Message consumers should never use the `PooledConnectionFactory` class.
Always Use the Failover Transport to Connect to Multiple Broker Endpoints

If you need your application to connect to multiple broker endpoints—for example, when you use an active/standby (p. 11) deployment mode or when you migrate from an on-premises message broker to Amazon MQ (p. 88)—use the Failover Transport to allow your consumers to randomly connect to either one. For example:

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

Avoid Using Message Selectors

It is possible to use JMS selectors to attach filters to topic subscriptions (to route messages to consumers based on their content). However, the use of JMS selectors fills up the Amazon MQ broker’s filter buffer, preventing it from filtering messages.

In general, avoid letting consumers route messages because, for optimal decoupling of consumers and producers, both the consumer and the producer should be ephemeral.

Prefer Virtual Destinations to Durable Subscriptions

A durable subscription can help ensure that the consumer receives all messages published to a topic, for example, after a lost connection is restored. However, the use of durable subscriptions also precludes the use of competing consumers and might have performance issues at scale. Consider using virtual destinations instead.

Ensuring Effective Amazon MQ Performance

The following design patterns can improve the effectiveness and performance of your Amazon MQ broker.

**Topics**
- Disable Concurrent Store and Dispatch for Queues with Slow Consumers (p. 97)
- Choose the Correct Broker Instance Type for the Best Throughput (p. 97)
Disable Concurrent Store and Dispatch for Queues with Slow Consumers

By default, Amazon MQ optimizes for queues with fast consumers:

- Consumers are considered fast if they are able to keep up with the rate of messages generated by producers.
- Consumers are considered slow if a queue builds up a backlog of unacknowledged messages, potentially causing a decrease in producer throughput.

To instruct Amazon MQ to optimize for queues with slow consumers, set the `concurrentStoreAndDispatchQueues` attribute to `false`. For an example configuration, see `concurrentStoreAndDispatchQueues` (p. 76).

Choose the Correct Broker Instance Type for the Best Throughput

The message throughput of a broker instance type (p. 39) depends on your application's use case and the following factors:

- Use of ActiveMQ in persistent mode
- Message size
- The number of producers and consumers
- The number of destinations

Understanding the Relationship Between Message Size, Latency, and Throughput

Depending on your use case, a larger broker instance type might not necessarily improve system throughput. When ActiveMQ writes messages to durable storage, the size of your messages determines your system's limiting factor:

- If your messages are smaller than 100 KB, persistent storage latency is the limiting factor.
- If your messages are larger than 100 KB, persistent storage throughput is the limiting factor.

When you use ActiveMQ in persistent mode, writing to storage normally occurs when there are either few consumers or when the consumers are slow. In non-persistent mode, writing to storage also occurs with slow consumers if the heap memory of the broker instance is full.

To determine the best broker instance type for your application, we recommend testing different broker instance types. For more information, see Broker Instance Types (p. 39) and also Measuring the Throughput for Amazon MQ using the JMS Benchmark.

Use Cases for Larger Broker Instance Types

There are three common use cases when larger broker instance types improve throughput:
• **Non-persistent mode** – When your application is less sensitive to losing messages during broker instance failover (p. 49) (for example, when broadcasting sports scores), you can often use ActiveMQ’s non-persistent mode. In this mode, ActiveMQ writes messages to persistent storage only if the heap memory of the broker instance is full. Systems that use non-persistent mode can benefit from the higher amount of memory, faster CPU, and faster network available on larger broker instance types.

• **Fast consumers** – When active consumers are available and the `concurrentStoreAndDispatchQueues` (p. 76) flag is enabled, ActiveMQ allows messages to flow directly from producer to consumer without sending messages to storage (even in persistent mode). If your application can consume messages quickly (or if you can design your consumers to do this), your application can benefit from a larger broker instance type. To let your application consume messages more quickly, add consumer threads to your application instances or scale up your application instances vertically or horizontally.

• **Batched transactions** – When you use persistent mode and send multiple messages per transaction, you can achieve an overall higher message throughput by using larger broker instance types. For more information, see Should I Use Transactions? in the ActiveMQ documentation.

---

### Choose the Correct Broker Storage Type for the Best Throughput

To take advantage of high durability and replication across multiple Availability Zones, use Amazon EFS. To take advantage of low latency and high throughput, use Amazon EBS. For more information, see Broker Storage (p. 44).

### Configure Your Network of Brokers Correctly

When you create a network of brokers (p. 50), configure it correctly for your application:

• **Enable persistent mode** – Because (relative to its peers) each broker instance acts like a producer or a consumer, networks of brokers don’t provide distributed replication of messages. The first broker that acts as a consumer receives a message and persists it to storage. This broker sends an acknowledgement to the producer and forwards the message to the next broker. When the second broker acknowledges the persistence of the message, the first broker deletes the message.

  If persistent mode is disabled, the first broker acknowledges the producer without persisting the message to storage. For more information, see Replicated Message Store and What is the difference between persistent and non-persistent delivery? in the Apache ActiveMQ documentation.

• **Don’t disable advisory messages for broker instances** – For more information, see Advisory Message in the Apache ActiveMQ documentation.

• **Don’t use multicast broker discovery** – Amazon MQ doesn’t support broker discovery using multicast. For more information, see What is the difference between discovery, multicast, and zeroconf? in the Apache ActiveMQ documentation.

### Avoid Slow Restarts by Recovering Prepared XA Transactions

ActiveMQ supports distributed (XA) transactions. Knowing how ActiveMQ processes XA transactions can help avoid slow recovery times for broker restarts and failovers in Amazon MQ.

Unresolved prepared XA transactions are replayed on every restart. If these remain unresolved, their number will grow over time, significantly increasing the time needed to start up the broker. This affects
Avoid Slow Restarts by Recovering Prepared XA Transactions

restart and failover time. You must resolve these transactions with a commit() or a rollback() so that performance doesn't degrade over time.

One cause of these unresolved transactions is an issue with Apache ActiveMQ. This may cause unresolved prepared transactions when Amazon MQ restarts. For more information, see the related Apache ActiveMQ defect.

To monitor your unresolved prepared XA transactions, you can use the JournalFilesForFastRecovery metric in Amazon CloudWatch Logs. If this number is increasing, or is consistently higher than 1, you should recover your unresolved transactions with code similar to the following example. For more information, see Quotas in Amazon MQ (p. 100).

The following example code walks through prepared XA transactions and closes them with a rollback().

```java
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) { 

        }
    }
}
```

In a real-world scenario, you could check your prepared XA transactions against your XA Transaction Manager. Then you can decide whether to handle each prepared transaction with a rollback() or a commit().
Quotas in Amazon MQ

This topic lists quotas within Amazon MQ. Many of the following quotas can be changed for specific AWS accounts. To request an increase for a limit, see AWS Service Quotas in the Amazon Web Services General Reference.

Topics
- Brokers (p. 100)
- Configurations (p. 100)
- Users (p. 101)
- Data Storage (p. 101)
- API Throttling (p. 102)

Brokers

The following table lists quotas related to Amazon MQ brokers.

<table>
<thead>
<tr>
<th>Limit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Broker name</td>
<td>• Must be unique in your AWS account.</td>
</tr>
<tr>
<td></td>
<td>• Must be 1-50 characters long.</td>
</tr>
<tr>
<td></td>
<td>• Must contain only characters specified in the ASCII Printable Character Set.</td>
</tr>
<tr>
<td></td>
<td>• Can contain only alphanumeric characters, dashes, periods, underscores, and tildes (- . _ ~).</td>
</tr>
<tr>
<td>Brokers per AWS account, per region</td>
<td>20</td>
</tr>
<tr>
<td>Broker configuration history depth</td>
<td>10</td>
</tr>
<tr>
<td>Connections per wire-level protocol</td>
<td>1,000 (100 for mq.t2.micro brokers)</td>
</tr>
<tr>
<td>Security groups per broker</td>
<td>5</td>
</tr>
<tr>
<td>Destinations (queues and topics) monitored in CloudWatch</td>
<td>CloudWatch monitors only the first 200 destinations.</td>
</tr>
</tbody>
</table>

Configurations

The following table lists quotas related to Amazon MQ configurations.

<table>
<thead>
<tr>
<th>Limit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration name</td>
<td>• Must be 1-150 characters long.</td>
</tr>
<tr>
<td></td>
<td>• Must contain only characters specified in the ASCII Printable Character Set.</td>
</tr>
</tbody>
</table>


### Users

The following table lists quotas related to Amazon MQ users.

<table>
<thead>
<tr>
<th>Limit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Username</td>
<td>• Must be 1-100 characters long.</td>
</tr>
<tr>
<td></td>
<td>• Must contain only characters specified in the ASCII Printable Character Set.</td>
</tr>
<tr>
<td></td>
<td>• Can contain only alphanumeric characters, dashes, periods, underscores, and tildes (- . _ ~).</td>
</tr>
<tr>
<td></td>
<td>• Must not contain commas (,).</td>
</tr>
<tr>
<td>Password</td>
<td>• Must be 12-250 characters long.</td>
</tr>
<tr>
<td></td>
<td>• Must contain only characters specified in the ASCII Printable Character Set.</td>
</tr>
<tr>
<td></td>
<td>• Must contain at least 4 unique characters.</td>
</tr>
<tr>
<td></td>
<td>• Must not contain commas (,).</td>
</tr>
<tr>
<td>Users per broker</td>
<td>250</td>
</tr>
<tr>
<td>Groups per user</td>
<td>20</td>
</tr>
</tbody>
</table>

### Data Storage

The following table lists quotas related to Amazon MQ data storage.

<table>
<thead>
<tr>
<th>Limit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage capacity per new mq.t2.micro broker. See Broker Instance Types (p. 39).</td>
<td>20 GB</td>
</tr>
<tr>
<td>Storage capacity per broker for other instance types. See Broker Instance Types (p. 39).</td>
<td>200 GB</td>
</tr>
<tr>
<td>Storage for scheduled jobs (JobSchedulerUsage) that are backed by Amazon EBS (p. 44)</td>
<td>50 GB</td>
</tr>
<tr>
<td>Temporary storage for brokers.</td>
<td>50 GB</td>
</tr>
</tbody>
</table>
API Throttling

The following throttling quotas are aggregated per AWS account, across all Amazon MQ APIs to maintain service bandwidth. For more information about Amazon MQ APIs, see the Amazon MQ REST API Reference.

**Important**
These quotas don't apply to ActiveMQ broker messaging APIs. For example, Amazon MQ doesn't throttle the sending or receiving of messages.

<table>
<thead>
<tr>
<th>Bucket Size</th>
<th>Refill Rate per Second</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>15</td>
</tr>
</tbody>
</table>
Monitoring and Logging Amazon MQ Brokers

This section provides information about monitoring and logging Amazon MQ broker activity.

Topics
- Monitoring Amazon MQ Brokers Using Amazon CloudWatch (p. 103)
- Logging Amazon MQ API Calls Using AWS CloudTrail (p. 107)
- Configuring Amazon MQ to Publish General and Audit Logs to Amazon CloudWatch Logs (p. 110)

Monitoring Amazon MQ Brokers Using Amazon CloudWatch

Amazon MQ and Amazon CloudWatch are integrated so you can use CloudWatch to view and analyze metrics for your ActiveMQ broker and the broker’s destinations (queues and topics). You can view and analyze your Amazon MQ metrics from the CloudWatch console, the AWS CLI, or the CloudWatch CLI. CloudWatch metrics for Amazon MQ are automatically polled from the broker and then pushed to CloudWatch every minute.

For information, see Accessing CloudWatch Metrics for Amazon MQ (p. 33).

Note
The following statistics are valid for all of the metrics:

- Average
- Minimum
- Maximum
- Sum

The AWS/AmazonMQ namespace includes the following metrics.

Broker Metrics

<table>
<thead>
<tr>
<th>Metric</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BurstBalance</td>
<td>Percent</td>
<td>The percentage of burst credits remaining on the Amazon EBS volume used to persist message data for throughput-optimized brokers. If this balance reaches zero, the IOPS provided by the Amazon EBS volume will decrease until the Burst Balance refills. For more information on how Burst Balances work in Amazon EBS, see: I/O Credits and Burst Performance.</td>
</tr>
<tr>
<td>Metric</td>
<td>Unit</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| CpuCreditBalance               | Credits (vCPU-minutes)        | **Important**
This metric is available only for the mq.t2.micro broker instance type.
CPU credit metrics are available only at five-minute intervals.
The number of earned CPU credits that an instance has accrued since it was launched or started (including the number of launch credits). The credit balance is available for the broker instance to spend on bursts beyond the baseline CPU utilization.
Credits are accrued in the credit balance after they're earned and removed from the credit balance after they're spent. The credit balance has a maximum limit. Once the limit is reached, any newly earned credits are discarded.                                                                 |
| CpuUtilization                 | Percent                       | The percentage of allocated Amazon EC2 compute units that the broker currently uses.                                                                                                                        |
| CurrentConnectionsCount        | Count                         | The current number of active connections on the current broker.                                                                                                                                             |
| EstablishedConnectionsCount    | Count                         | The total number of connections, active and inactive, that have been established on the broker.                                                                                                             |
| HeapUsage                      | Percent                       | The percentage of the ActiveMQ JVM memory limit that the broker currently uses.                                                                                                                               |
| InactiveDurableTopicSubscribers| Count                         | The number of inactive durable topic subscribers, up to a maximum of 2000.                                                                                                                                |
| JobSchedulerStorePercentUsage  | Percent                       | The percentage of disk space used by the job scheduler store.                                                                                                                                                |
| JournalFilesForFastRecovery    | Count                         | The number of journal files that will be replayed after a clean shutdown.                                                                                                                                   |
### Broker Metrics

<table>
<thead>
<tr>
<th>Metric</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JournalFilesForFullRecoveryCount</td>
<td>Count</td>
<td>The number of journal files that will be replayed after an unclean shutdown.</td>
</tr>
<tr>
<td>NetworkIn</td>
<td>Bytes</td>
<td>The volume of incoming traffic for the broker.</td>
</tr>
<tr>
<td>NetworkOut</td>
<td>Bytes</td>
<td>The volume of outgoing traffic for the broker.</td>
</tr>
<tr>
<td>OpenTransactionCount</td>
<td>Count</td>
<td>The total number of transactions in progress.</td>
</tr>
<tr>
<td>StorePercentUsage</td>
<td>Percent</td>
<td>The percent used by the storage limit. If this reaches 100, the broker will refuse messages.</td>
</tr>
<tr>
<td>TempPercentUsage</td>
<td>Percent</td>
<td>The percentage of available temporary storage used by non-persistent messages.</td>
</tr>
<tr>
<td>TotalConsumerCount</td>
<td>Count</td>
<td>The number of message consumers subscribed to destinations on the current broker.</td>
</tr>
<tr>
<td>TotalMessageCount</td>
<td>Count</td>
<td>The number of messages stored on the broker.</td>
</tr>
<tr>
<td>TotalProducerCount</td>
<td>Count</td>
<td>The number of message producers active on destinations on the current broker.</td>
</tr>
<tr>
<td>VolumeReadOps</td>
<td>Count</td>
<td>The number of read operations performed on the Amazon EBS volume.</td>
</tr>
<tr>
<td>VolumeWriteOps</td>
<td>Count</td>
<td>The number of write operations performed on the Amazon EBS volume.</td>
</tr>
</tbody>
</table>

### Dimension for Broker Metrics

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Broker</td>
<td>The name of the broker.</td>
</tr>
</tbody>
</table>

**Note**
A single-instance broker has the suffix -1. An active/standby broker for high availability has the suffixes -1 and -2 for its redundant pair.
Destination (Queue and Topic) Metrics

**Important**
The following metrics include per-minute counts for the CloudWatch polling period.

- EnqueueCount
- ExpiredCount
- DequeueCount
- DispatchCount
- InFlightCount

For example, in a five-minute CloudWatch period, EnqueueCount has five count values, each for a one-minute portion of the period. The Minimum and Maximum statistics provide the lowest and highest per-minute value during the specified period.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConsumerCount</td>
<td>Count</td>
<td>The number of consumers subscribed to the destination.</td>
</tr>
<tr>
<td>EnqueueCount</td>
<td>Count</td>
<td>The number of messages sent to the destination, per minute.</td>
</tr>
<tr>
<td>EnqueueTime</td>
<td>Time (milliseconds)</td>
<td>The end-to-end latency from when a message arrives at a broker until it is delivered to a consumer.</td>
</tr>
<tr>
<td>ExpiredCount</td>
<td>Count</td>
<td>The number of messages that couldn't be delivered because they expired, per minute.</td>
</tr>
<tr>
<td>DispatchCount</td>
<td>Count</td>
<td>The number of messages sent to consumers, per minute.</td>
</tr>
<tr>
<td>DequeueCount</td>
<td>Count</td>
<td>The number of messages acknowledged by consumers, per minute.</td>
</tr>
<tr>
<td>InFlightCount</td>
<td>Count</td>
<td>The number of messages sent to consumers that have not been acknowledged.</td>
</tr>
<tr>
<td>ReceiveCount</td>
<td>Count</td>
<td>The number of messages that have been received from the remote broker for a duplex network connector.</td>
</tr>
<tr>
<td>MemoryUsage</td>
<td>Percent</td>
<td>The percentage of the memory limit that the destination currently uses.</td>
</tr>
<tr>
<td>ProducerCount</td>
<td>Count</td>
<td>The number of producers for the destination.</td>
</tr>
<tr>
<td>QueueSize</td>
<td>Count</td>
<td>The number of messages in the queue.</td>
</tr>
<tr>
<td>Metric</td>
<td>Unit</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TotalEnqueueCount</td>
<td>Count</td>
<td>The total number of messages that have been sent to the broker.</td>
</tr>
<tr>
<td>TotalDequeueCount</td>
<td>Count</td>
<td>The total number of messages that have been consumed by clients.</td>
</tr>
</tbody>
</table>

**Important**
This metric applies only to queues.

**Note**
TotalEnqueueCount and TotalDequeueCount metrics include messages for advisory topics. For more information about advisory topic messages, see the [ActiveMQ documentation](#).

**Dimensions for Destination (Queue and Topic) Metrics**

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Broker</td>
<td>The name of the broker.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong> A single-instance broker has the suffix -1. An active/standby broker for high availability has the suffixes -1 and -2 for its redundant pair.</td>
</tr>
<tr>
<td>Topic or Queue</td>
<td>The name of the topic or queue.</td>
</tr>
<tr>
<td>NetworkConnector</td>
<td>The name of the network connector.</td>
</tr>
</tbody>
</table>

**Logging Amazon MQ API Calls Using AWS CloudTrail**

Amazon MQ is integrated with AWS CloudTrail, a service that provides a record of the Amazon MQ calls that a user, role, or AWS service makes. CloudTrail captures API calls related to Amazon MQ brokers and configurations as events, including calls from the Amazon MQ console and code calls from Amazon MQ APIs. For more information about CloudTrail, see the [AWS CloudTrail User Guide](#).

**Note**
CloudTrail doesn't log API calls related to ActiveMQ operations (for example, sending and receiving messages) or to the ActiveMQ Web Console. To log information related to ActiveMQ operations, you can configure Amazon MQ to publish general and audit logs to Amazon CloudWatch Logs (p. 110).

Using the information that CloudTrail collects, you can identify a specific request to an Amazon MQ API, the IP address of the requester, the requester's identity, the date and time of the request, and so on. If you configure a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket. If you don't configure a trail, you can view the most recent events in the event history in the CloudTrail console. For more information, see [Overview for Creating a Trail](#) in the [AWS CloudTrail User Guide](#).
Amazon MQ Information in CloudTrail

When you create your AWS account, CloudTrail is enabled. When a supported Amazon MQ event activity occurs, it is recorded in a CloudTrail event with other AWS service events in the event history. You can view, search, and download recent events for your AWS account. For more information, see Viewing Events with CloudTrail Event History in the AWS CloudTrail User Guide.

A trail allows CloudTrail to deliver log files to an Amazon S3 bucket. You can create a trail to keep an ongoing record of events in your AWS account. By default, when you create a trail using the AWS Management Console, the trail applies to all AWS Regions. The trail logs events from all AWS Regions and delivers log files to the specified Amazon S3 bucket. You can also configure other AWS services to further analyze and act on the event data collected in CloudTrail logs. For more information, see the following topics in the AWS CloudTrail User Guide:

- CloudTrail Supported Services and Integrations
- Configuring Amazon SNS Notifications for CloudTrail
- Receiving CloudTrail Log Files from Multiple Regions
- Receiving CloudTrail Log Files from Multiple Accounts

Amazon MQ supports logging both the request parameters and the responses for the following APIs as events in CloudTrail log files:

- CreateConfiguration
- DeleteBroker
- DeleteUser
- RebootBroker
- UpdateBroker

**Important**

For the GET methods of the following APIs, the request parameters are logged, but the responses are redacted:

- DescribeBroker
- DescribeConfiguration
- DescribeConfigurationRevision
- DescribeUser
- ListBrokers
- ListConfigurationRevisions
- ListConfigurations
- ListUsers

For the following APIs, the data and password request parameters are hidden by asterisks (***):

- CreateBroker (POST)
- CreateUser (POST)
- UpdateConfiguration (PUT)
- UpdateUser (PUT)
Every event or log entry contains information about the requester. This information helps you determine the following:

- Was the request made with root or IAM user credentials?
- Was the request made with temporary security credentials for a role or a federated user?
- Was the request made by another AWS service?

For more information, see CloudTrail userIdentity Element in the AWS CloudTrail User Guide.

Example Amazon MQ Log File Entry

A trail is a configuration that allows the delivery of events as log files to the specified Amazon S3 bucket. CloudTrail log files contain one or more log entries.

An event represents a single request from any source and includes information about the request to an Amazon MQ API, the IP address of the requester, the requester's identity, the date and time of the request, and so on.

The following example shows a CloudTrail log entry for a CreateBroker API call.

Note
Because CloudTrail log files aren't an ordered stack trace of public APIs, they don't list information in any specific order.

```json
{
  "eventVersion": "1.06",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AKIAIOSFODNN7EXAMPLE",
    "arn": "arn:aws:iam::111122223333:user/AmazonMqConsole",
    "accountId": "111122223333",
    "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
    "userName": "AmazonMqConsole"
  },
  "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"
        ]
      }
    ]
  }
}
```
Configuring Amazon MQ to Publish General and Audit Logs to Amazon CloudWatch Logs

Amazon MQ is integrated with Amazon CloudWatch Logs, a service that monitors, stores, and accesses your log files from a variety of sources. For example, you can configure CloudWatch alarms to receive notifications of broker reboots or troubleshoot broker configuration (p. 61) errors. For more information about CloudWatch Logs, see the Amazon CloudWatch Logs User Guide.

To allow Amazon MQ to publish logs to CloudWatch Logs, you must add a permission to your Amazon MQ user (p. 111) and also configure a resource-based policy for Amazon MQ (p. 112) before you create or restart the broker.

For more information about configuring Amazon MQ to publish general and audit logs to CloudWatch Logs, see Configure Advanced Broker Settings (p. 12).

Topics

- Understanding the Structure of Logging in CloudWatch Logs (p. 111)
- Add the CreateLogGroup Permission to Your Amazon MQ User (p. 111)
- Configure a Resource-Based Policy for Amazon MQ (p. 112)
- Troubleshooting CloudWatch Logs Configuration (p. 112)
Understanding the Structure of Logging in CloudWatch Logs

You can enable general and audit logging when you configure advanced broker settings (p. 12) when you create a broker, or when you edit a broker.

General logging enables the default INFO logging level (DEBUG logging isn't supported) and publishes activemq.log to a log group in your CloudWatch account. The log group has a format similar to the following:

/aws/amazonmq/broker/b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9/general

Audit logging enables logging of management actions taken using JMX or using the ActiveMQ Web Console and publishes audit.log to a log group in your CloudWatch account. The log group has a format similar to the following:

/aws/amazonmq/broker/b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9/audit

Depending on whether you have a single-instance broker (p. 46) or an active/standby broker (p. 49), Amazon MQ creates either one or two log streams within each log group. The log streams have a format similar to the following.

activemq-b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-1.log
activemq-b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9-2.log

The -1 and -2 suffixes denote individual broker instances. For more information, see Working with Log Groups and Log Streams in the Amazon CloudWatch Logs User Guide.

Add the CreateLogGroup Permission to Your Amazon MQ User

To allow Amazon MQ to create a CloudWatch Logs log group, you must ensure that the user who creates or reboots the broker has the logs:CreateLogGroup permission.

Important
If you don't add the CreateLogGroup permission to your Amazon MQ user before the user creates or reboots the broker, Amazon MQ doesn't create the log group.

The following example IAM-based policy grants permission for logs:CreateLogGroup to user 111122223333.

```json
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Principal": {
            "AWS": "111122223333"
         },
         "Action": "logs:CreateLogGroup",
         "Resource": "arn:aws:logs:*::log-group:/aws/amazonmq/*"
      }
   ]
}
```
Configure a Resource-Based Policy for Amazon MQ

To allow Amazon MQ to publish logs to your CloudWatch Logs log group, configure a resource-based policy to give Amazon MQ access to the following CloudWatch Logs API actions:

- **CreateLogStream** – Creates a CloudWatch Logs log stream for the specified log group.
- **PutLogEvents** – Delivers events to the specified CloudWatch Logs log stream.

**Important**

If you don’t configure a resource-based policy for Amazon MQ, the broker can’t publish the logs to CloudWatch Logs.

The following example resource-based policy grants permission for **logs/CreateLogStream** and **logs/PutLogEvents** to 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/*"
}
]
}
```

**Note**

Because this example uses the `/aws/amazonmq/` prefix, you need to configure the resource-based policy only once per AWS account, per region.

You can achieve the same effect using the following AWS CLI command:

```
```

Troubleshooting CloudWatch Logs Configuration

In some cases, CloudWatch Logs might not always behave as expected. This section gives an overview of common issues and shows how to resolve them.

**Log Groups Don’t Appear in CloudWatch**

Add the **CreateLogGroup** permission to your Amazon MQ user (p. 111) and reboot the broker. This allows Amazon MQ to create the log group.
Log Streams Don't Appear in CloudWatch Log Groups

Configure a resource-based policy for Amazon MQ (p. 112). This allows your broker to publish its logs.
Amazon MQ Security

This section provides information about Amazon MQ and ActiveMQ authentication and authorization. For information about security best practices, see Using Amazon MQ Securely (p. 93).

Topics

• Tag-based Policies (p. 114)
• Authentication (p. 114)
• Encryption (p. 116)
• API Authentication and Authorization for Amazon MQ (p. 117)
• Messaging Authentication and Authorization for ActiveMQ (p. 120)

Tag-based Policies

Amazon MQ supports policies based on tags. For instance, you could deny access to Amazon MQ resources that include a tag with the key environment and the value production:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "mq:DeleteBroker",
                "mq:RebootBroker",
                "mq:DeleteTag"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/environment": "production"
                }
            }
        }
    ]
}
```

This policy will Deny the ability to delete or reboot an Amazon MQ broker that includes the tag environment/production.

For more information on tagging, see:

• Tagging resources (p. 85)
• Controlling Access Using IAM Tags

Authentication

You can access AWS as any of the following types of identities:

• **AWS account root user** – When you first create an AWS account, you begin with a single sign-in identity that has complete access to all AWS services and resources in the account. This identity is
called the AWS account root user and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you do not use the root user for your everyday tasks, even the administrative ones. Instead, adhere to the best practice of using the root user only to create your first IAM user. Then securely lock away the root user credentials and use them to perform only a few account and service management tasks.

- **IAM user** – An IAM user is an identity within your AWS account that has specific custom permissions (for example, permissions to create a broker in Amazon MQ). You can use an IAM user name and password to sign in to secure AWS webpages like the AWS Management Console, AWS Discussion Forums, or the AWS Support Center.

In addition to a user name and password, you can also generate access keys for each user. You can use these keys when you access AWS services programmatically, either through one of the several SDKs or by using the AWS Command Line Interface (CLI). The SDK and CLI tools use the access keys to cryptographically sign your request. If you don't use AWS tools, you must sign the request yourself. Amazon MQ supports Signature Version 4, a protocol for authenticating inbound API requests. For more information about authenticating requests, see Signature Version 4 Signing Process in the AWS General Reference.

- **IAM role** – An IAM role is an IAM identity that you can create in your account that has specific permissions. An IAM role is similar to an IAM user in that it is an AWS identity with permissions policies that determine what the identity can and cannot do in AWS. However, instead of being uniquely associated with one person, a role is intended to be assumable by anyone who needs it. Also, a role does not have standard long-term credentials such as a password or access keys associated with it. Instead, when you assume a role, it provides you with temporary security credentials for your role session. IAM roles with temporary credentials are useful in the following situations:

- **Federated user access** – Instead of creating an IAM user, you can use existing identities from AWS Directory Service, your enterprise user directory, or a web identity provider. These are known as federated users. AWS assigns a role to a federated user when access is requested through an identity provider. For more information about federated users, see Federated Users and Roles in the IAM User Guide.

- **AWS service access** – A service role is an IAM role that a service assumes to perform actions in your account on your behalf. When you set up some AWS service environments, you must define a role for the service to assume. This service role must include all the permissions that are required for the service to access the AWS resources that it needs. Service roles vary from service to service, but many allow you to choose your permissions as long as you meet the documented requirements for that service. Service roles provide access only within your account and cannot be used to grant access to services in other accounts. You can create, modify, and delete a service role from within IAM. For example, you can create a role that allows Amazon Redshift to access an Amazon S3 bucket on your behalf and then load data from that bucket into an Amazon Redshift cluster. For more information, see Creating a Role to Delegate Permissions to an AWS Service in the IAM User Guide.

- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances in the IAM User Guide.
Encryption

User data stored in Amazon MQ is encrypted at rest. Amazon MQ encryption at rest provides enhanced security by encrypting your data using encryption keys stored in the AWS Key Management Service (KMS). This service helps reduce the operational burden and complexity involved in protecting sensitive data. With encryption at rest, you can build security-sensitive applications that meet encryption compliance and regulatory requirements.

All connections between Amazon MQ brokers use Transport layer Security (TLS) to provide encryption in transit.

Amazon MQ encrypts messages at rest and in transit using encryption keys that it manages and stores securely. For additional security, we highly recommend designing your application to use client-side encryption. For more information, see the AWS Encryption SDK Developer Guide.

Encryption at Rest

Amazon MQ integrates with AWS Key Management Service (KMS) to offer transparent server-side encryption. Amazon MQ always encrypts your data at rest. When you create a broker, you can specify the AWS KMS customer master key (CMK) that you want Amazon MQ to use to encrypt your data at rest. If you don't specify a CMK, Amazon MQ creates an AWS managed CMK for you and uses it on your behalf. For more information about CMKs, see Customer Master Keys (CMKs) in the AWS Key Management Service Developer Guide.

When creating a broker, you can configure what Amazon MQ uses for your encryption key by selecting one of the following.

- **AWS owned CMK** — The key is owned by Amazon MQ and is not in your account.
- **AWS managed CMK** — The AWS managed CMK (aws/mq) is a CMK in your account that is created, managed, and used on your behalf by Amazon MQ.
- **Select existing customer managed CMK** — Customer managed CMKs are created and managed by you in AWS Key Management Service (KMS).

**Important**

Amazon MQ uses Amazon Elastic File System (EFS) to store message data. If you revoke the grant that gives Amazon EFS permission to use the KMS keys in your account, Amazon MQ cannot access this data and your broker will stop working. When you revoke a grant for Amazon EFS, it will not take place immediately. To revoke access rights, delete your broker rather than revoking the grant.

Encryption in Transit

Amazon MQ encrypts data in transit between the brokers of your Amazon MQ deployment. All data that passes between Amazon MQ brokers is encrypted using Transport layer Security (TLS). This is true for all available protocols.

You can access your brokers using the following protocols with TLS enabled:

- AMQP
- MQTT
- MQTT over WebSocket
- OpenWire
- STOMP
API Authentication and Authorization for Amazon MQ

Amazon MQ uses standard AWS request signing for API authentication. For more information, see Signing AWS API Requests in the AWS General Reference.

**Note**
Currently, Amazon MQ doesn't support IAM authentication using resource-based permissions or resource-based policies.

To authorize AWS users to work with brokers, configurations, and users, you must edit your IAM policy permissions.

**Topics**
- IAM Permissions Required to Create an Amazon MQ Broker (p. 117)
- Amazon MQ REST API Permissions Reference (p. 118)
- Resource-Level Permissions for Amazon MQ API Actions (p. 119)

### IAM Permissions Required to Create an Amazon MQ Broker

To create a broker, you must either use the `AmazonMQFullAccess` IAM policy or include the following EC2 permissions in your IAM policy.

The following custom policy is comprised of two statements (one conditional) which grant permissions to manipulate the resources which Amazon MQ requires to create an ActiveMQ broker.

**Important**
- The `ec2:CreateNetworkInterface` action is required to allow Amazon MQ to create an elastic network interface (ENI) in your account on your behalf.
- The `ec2:CreateNetworkInterfacePermission` action authorizes Amazon MQ to attach the ENI to an ActiveMQ broker.
- The `ec2:AuthorizedService` condition key ensures that ENI permissions can be granted only to Amazon MQ service accounts.

```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"
  }
}

}

For more information, see Create an IAM User and Get Your AWS Credentials (p. 3) and Never Modify or Delete the Amazon MQ Elastic Network Interface (p. 94).

**Amazon MQ REST API Permissions Reference**

The following table lists Amazon MQ REST APIs and the corresponding IAM permissions.

### Amazon MQ REST APIs and Required Permissions

<table>
<thead>
<tr>
<th>Amazon MQ REST APIs</th>
<th>Required Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateBroker</td>
<td>mq:CreateBroker</td>
</tr>
<tr>
<td>CreateConfiguration</td>
<td>mq:CreateConfiguration</td>
</tr>
<tr>
<td>CreateTags</td>
<td>mg:CreateTags</td>
</tr>
<tr>
<td>CreateUser</td>
<td>mq:CreateUser</td>
</tr>
<tr>
<td>DeleteBroker</td>
<td>mq:DeleteBroker</td>
</tr>
<tr>
<td>DeleteUser</td>
<td>mq:DeleteUser</td>
</tr>
<tr>
<td>DescribeBroker</td>
<td>mq:DescribeBroker</td>
</tr>
<tr>
<td>DescribeConfiguration</td>
<td>mq:DescribeConfiguration</td>
</tr>
<tr>
<td>DescribeConfigurationRevision</td>
<td>mq:DescribeConfigurationRevision</td>
</tr>
<tr>
<td>DescribeUser</td>
<td>mq:DescribeUser</td>
</tr>
<tr>
<td>ListBrokers</td>
<td>mq:ListBrokers</td>
</tr>
<tr>
<td>ListConfigurationRevisions</td>
<td>mq:ListConfigurationRevisions</td>
</tr>
<tr>
<td>ListConfigurations</td>
<td>mq:ListConfigurations</td>
</tr>
<tr>
<td>ListTags</td>
<td>mq:ListTags</td>
</tr>
<tr>
<td>ListUsers</td>
<td>mq:ListUsers</td>
</tr>
<tr>
<td>RebootBroker</td>
<td>mq:RebootBroker</td>
</tr>
<tr>
<td>UpdateBroker</td>
<td>mq:UpdateBroker</td>
</tr>
</tbody>
</table>
Resource-Level Permissions for Amazon MQ API Actions

The term resource-level permissions refers to the ability to specify the resources on which users are allowed to perform actions. Amazon MQ has partial support for resource-level permissions. For certain Amazon MQ actions, you can control when users are allowed to use those actions based on conditions that have to be fulfilled, or specific resources that users are allowed to use.

The following table describes the Amazon MQ API actions that currently support resource-level permissions, as well as the supported resources, resource ARNs, and condition keys for each action.

**Important**

If an Amazon MQ API action is not listed in this table, then it does not support resource-level permissions. If an Amazon MQ API action does not support resource-level permissions, you can grant users permission to use the action, but you have to specify a * wildcard for the resource element of your policy statement.

<table>
<thead>
<tr>
<th>API Action</th>
<th>Resource Types (*required)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateConfiguration</td>
<td>configurations*</td>
</tr>
<tr>
<td>CreateTags</td>
<td>brokers, configurations</td>
</tr>
<tr>
<td>CreateUser</td>
<td>brokers*</td>
</tr>
<tr>
<td>DeleteBroker</td>
<td>brokers*</td>
</tr>
<tr>
<td>DeleteUser</td>
<td>brokers*</td>
</tr>
<tr>
<td>DescribeBroker</td>
<td>brokers*</td>
</tr>
<tr>
<td>DescribeConfiguration</td>
<td>configurations*</td>
</tr>
<tr>
<td>DescribeConfigurationRevision</td>
<td>configurations*</td>
</tr>
<tr>
<td>DescribeUser</td>
<td>brokers*</td>
</tr>
<tr>
<td>ListConfigurationRevisions</td>
<td>configurations*</td>
</tr>
<tr>
<td>ListTags</td>
<td>brokers, configurations</td>
</tr>
<tr>
<td>ListUsers</td>
<td>brokers*</td>
</tr>
<tr>
<td>RebootBroker</td>
<td>brokers*</td>
</tr>
<tr>
<td>UpdateBroker</td>
<td>brokers*</td>
</tr>
<tr>
<td>UpdateConfiguration</td>
<td>configurations*</td>
</tr>
<tr>
<td>UpdateUser</td>
<td>brokers*</td>
</tr>
</tbody>
</table>
Messaging Authentication and Authorization for ActiveMQ

You can access your brokers using the following protocols with TLS enabled:

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

Amazon MQ uses native ActiveMQ authentication to manage user permissions. For information about restrictions related to ActiveMQ usernames and passwords, see Limits Related to Users (p. 101).

To authorize ActiveMQ users and groups to work with queues and topics, you must edit your broker's configuration (p. 22). Amazon MQ uses ActiveMQ's Simple Authentication Plugin to restrict reading and writing to destinations. For more information and examples, see Always Configure an Authorization Map (p. 94) and authorizationEntry (p. 74).

Note
Currently, Amazon MQ doesn't support Client Certificate Authentication or plugins for Java Authentication and Authorization Service (JAAS).
Related Resources

Amazon MQ Resources

The following table lists useful resources for working with Amazon MQ.

<table>
<thead>
<tr>
<th>Resource</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon MQ REST API Reference</td>
<td>Descriptions of REST resources, example requests, HTTP methods, schemas, parameters, and the errors that the service returns.</td>
</tr>
<tr>
<td>Amazon MQ in the AWS CLI Command Reference</td>
<td>Descriptions of the AWS CLI commands that you can use to work with message brokers.</td>
</tr>
<tr>
<td>Amazon MQ in the AWS CloudFormation User Guide</td>
<td>The AWS::Amazon MQ::Broker resource lets you create Amazon MQ brokers, add configuration changes or modify users for the specified broker, return information about the specified broker, and delete the specified broker. The AWS::Amazon MQ::Configuration resource lets you create Amazon MQ configurations, add configuration changes or modify users, and return information about the specified configuration.</td>
</tr>
<tr>
<td>Regions and Endpoints</td>
<td>Information about Amazon MQ regions and endpoints</td>
</tr>
<tr>
<td>Product Page</td>
<td>The primary web page for information about Amazon MQ.</td>
</tr>
<tr>
<td>Discussion Forum</td>
<td>A community-based forum for developers to discuss technical questions related to Amazon MQ.</td>
</tr>
<tr>
<td>AWS Premium Support Information</td>
<td>The primary web page for information about AWS Premium Support, a one-on-one, fast-response support channel to help you build and run applications on AWS infrastructure services</td>
</tr>
</tbody>
</table>

Apache ActiveMQ Resources

The following table lists useful resources for working with Apache ActiveMQ.

<table>
<thead>
<tr>
<th>Resource</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache ActiveMQ Getting Started Guide</td>
<td>The official documentation of Apache ActiveMQ.</td>
</tr>
</tbody>
</table>
### Resource

<table>
<thead>
<tr>
<th>Resource</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>ActiveMQ in Action</em></td>
<td>A guide to Apache ActiveMQ that covers the anatomy of JMS messages, connectors, message persistence, authentication, and authorization.</td>
</tr>
<tr>
<td><em>Cross-Language Clients</em></td>
<td>A list of programming languages and corresponding Apache ActiveMQ libraries. See also <a href="http://amzn.to/2s2rLiE">ActiveMQ Client</a> and <a href="http://amzn.to/2s2rLiE">QpidJMS Client</a>.</td>
</tr>
</tbody>
</table>
Amazon MQ Release Notes

The following table lists Amazon MQ feature releases and improvements. For changes to the Amazon MQ Developer Guide, see Amazon MQ Document History (p. 130).

<table>
<thead>
<tr>
<th>Date</th>
<th>Documentation Update</th>
</tr>
</thead>
<tbody>
<tr>
<td>April 30, 2020</td>
<td>Amazon MQ supports a new child collection element, systemUsage, on the broker element. For more information, see systemUsage (p. 77). Amazon MQ also supports three new attributes on the kahaDB child element.</td>
</tr>
<tr>
<td></td>
<td>• journalDiskSyncInterval - Interval (ms) for when to perform a disk sync if journalDiskSyncStrategy=periodic. • journalDiskSyncStrategy - configures the disk sync policy. • preallocationStrategy - configures how the broker will try to preallocate the journal files when a new journal file is needed. For more information, see Attributes (p. 76).</td>
</tr>
<tr>
<td>March 3, 2020</td>
<td>Amazon MQ supports two new CloudWatch metrics</td>
</tr>
<tr>
<td></td>
<td>• TempPercentUsage - The percentage of available temporary storage used by non-persistent messages. • JobSchedulerStorePercentUsage - The percentage of disk space used by the job scheduler store. For more information, see Monitoring Amazon MQ Using CloudWatch (p. 103).</td>
</tr>
<tr>
<td>February 4, 2020</td>
<td>Amazon MQ is available in the Asia Pacific (Hong Kong) and Middle East (Bahrain) regions. For information on available regions, see AWS Regions and Endpoints.</td>
</tr>
<tr>
<td>January 22, 2020</td>
<td>Amazon MQ supports ActiveMQ 5.15.10. For more information, see the following:</td>
</tr>
<tr>
<td></td>
<td>• ActiveMQ 5.15.10 Release Notes • Broker Engine (p. 43) • Working with Spring XML Configuration Files (p. 61)</td>
</tr>
<tr>
<td>December 19, 2019</td>
<td>Amazon MQ is available in the Europe (Stockholm) and South America (São Paulo) regions. For information on available regions, see AWS Regions and Endpoints.</td>
</tr>
<tr>
<td>December 16, 2019</td>
<td>Amazon MQ supports creating throughput-optimized brokers by using Amazon Elastic Block Store (EBS)—instead of the default Amazon Elastic File System (Amazon EFS)—for broker storage. To take advantage of high durability and replication across multiple Availability Zones, use Amazon EFS. To take advantage of low latency and high throughput, use Amazon EBS. Important • You can use Amazon EBS only with the mq.m5 broker instance type family. • Although you can change the broker instance type, you can’t change the broker storage type after you create the broker.</td>
</tr>
<tr>
<td>Date</td>
<td>Documentation Update</td>
</tr>
<tr>
<td>-------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>October 18, 2019</td>
<td>Amazon EBS replicates data within a single Availability Zone and doesn't support the active/standby (p. 49) deployment mode. For more information, see the following:</td>
</tr>
<tr>
<td></td>
<td>• Broker Storage (p. 44)</td>
</tr>
<tr>
<td></td>
<td>• Choose the Correct Broker Storage Type for the Best Throughput (p. 98)</td>
</tr>
<tr>
<td></td>
<td>• The <code>storageType</code> property of the <code>broker-instance-options</code> resource in the <code>Amazon MQ REST API Reference</code></td>
</tr>
<tr>
<td></td>
<td>• The BurstBalance, VolumeReadOps, and VolumeWriteOps metrics in the Broker Metrics (p. 103) section.</td>
</tr>
<tr>
<td>October 11, 2019</td>
<td>Two Amazon CloudWatch metrics are available: <code>TotalEnqueueCount</code> and <code>TotalDequeueCount</code>. For more information, see Destination (Queue and Topic) Metrics (p. 106).</td>
</tr>
<tr>
<td>September 30, 2019</td>
<td>Amazon MQ now supports Federal Information Processing Standard 140-2 (FIPS) compliant endpoints in U.S. commercial regions. For more information see the following:</td>
</tr>
<tr>
<td></td>
<td>• Federal Information Processing Standard (FIPS) 140-2</td>
</tr>
<tr>
<td></td>
<td>• Amazon MQ Regions and Endpoints</td>
</tr>
<tr>
<td>August 30, 2019</td>
<td>You can now update the security groups associated with a broker, both in the console and with <code>UpdateBrokerInput</code>.</td>
</tr>
<tr>
<td>July 22, 2019</td>
<td>Amazon MQ integrates with AWS Key Management Service (KMS) to offer server-side encryption. You can now select your own customer managed CMK, or use an AWS managed CMK in your KMS account. For more information, see Encryption at Rest (p. 116).</td>
</tr>
<tr>
<td></td>
<td>Amazon MQ supports using customer master keys (CMK) in the following ways.</td>
</tr>
<tr>
<td></td>
<td>• <strong>AWS owned CMK</strong> — The key is owned Amazon MQ and is not in your account.</td>
</tr>
<tr>
<td></td>
<td>• <strong>AWS managed CMK</strong> — The AWS managed CMK (aws/mq) is a CMK in your account that is created, managed, and used on your behalf by Amazon MQ.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Select existing customer managed CMK</strong> — Customer managed CMKs are created and managed by you in AWS Key Management Service (KMS).</td>
</tr>
<tr>
<td>June 19, 2019</td>
<td>Amazon MQ is available in the Europe (Paris) and Asia Pacific (Mumbai) regions. For information on available regions, see AWS Regions and Endpoints.</td>
</tr>
<tr>
<td>June 12, 2019</td>
<td>Amazon MQ is available in the Canada (Central) region. For information on available regions, see AWS Regions and Endpoints.</td>
</tr>
<tr>
<td>Date</td>
<td>Documentation Update</td>
</tr>
<tr>
<td>--------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>June 3, 2019</td>
<td>Two new Amazon CloudWatch metrics are available: <code>EstablishedConnectionsCount</code> and <code>InactiveDurableSubscribers</code>. For more information, see the following:</td>
</tr>
<tr>
<td></td>
<td>• Monitoring Amazon MQ Using CloudWatch (p. 103)</td>
</tr>
<tr>
<td></td>
<td>• Broker Metrics (p. 103)</td>
</tr>
<tr>
<td>May 10, 2019</td>
<td>Data storage for new <code>mq.t2.micro</code> instance types is limited to 20 GB. For more information, see the following:</td>
</tr>
<tr>
<td></td>
<td>• Limits Related to Data Storage (p. 101)</td>
</tr>
<tr>
<td></td>
<td>• Broker Instance Types (p. 39)</td>
</tr>
<tr>
<td>April 29, 2019</td>
<td>You can now use tag-based policies and resource-level permissions. For more information, see the following:</td>
</tr>
<tr>
<td></td>
<td>• Tag-based Policies (p. 114)</td>
</tr>
<tr>
<td></td>
<td>• Resource-Level Permissions for Amazon MQ API Actions (p. 119)</td>
</tr>
<tr>
<td>April 16, 2019</td>
<td>You can now retrieve information about broker engine and broker instance options using the REST API. For more information, see the following:</td>
</tr>
<tr>
<td></td>
<td>• Broker Instance Options</td>
</tr>
<tr>
<td></td>
<td>• Broker Engine Types</td>
</tr>
<tr>
<td>April 8, 2019</td>
<td>Amazon MQ supports ActiveMQ 5.15.9. For more information, see the following:</td>
</tr>
<tr>
<td></td>
<td>• ActiveMQ 5.15.9 Release Notes</td>
</tr>
<tr>
<td></td>
<td>• Broker Engine (p. 43)</td>
</tr>
<tr>
<td></td>
<td>• Working with Spring XML Configuration Files (p. 61)</td>
</tr>
<tr>
<td>March 4, 2019</td>
<td>Improved the documentation for configuring dynamic failover and the rebalancing of clients for a network of brokers. Enable dynamic failover by configuring <code>transportConnectors</code> along with <code>networkConnectors</code> configuration options. For more information, see the following:</td>
</tr>
<tr>
<td></td>
<td>• Dynamic Failover With Transport Connectors (p. 59)</td>
</tr>
<tr>
<td></td>
<td>• Amazon MQ Network of Brokers (p. 50)</td>
</tr>
<tr>
<td></td>
<td>• Amazon MQ Broker Configuration Parameters (p. 61)</td>
</tr>
<tr>
<td>February 27, 2019</td>
<td>Amazon MQ is available in the Europe (London) Region in addition to the following regions:</td>
</tr>
<tr>
<td></td>
<td>• Asia Pacific (Singapore)</td>
</tr>
<tr>
<td></td>
<td>• US East (Ohio)</td>
</tr>
<tr>
<td></td>
<td>• US East (N. Virginia)</td>
</tr>
<tr>
<td></td>
<td>• US West (N. California)</td>
</tr>
<tr>
<td></td>
<td>• US West (Oregon)</td>
</tr>
<tr>
<td></td>
<td>• Asia Pacific (Tokyo)</td>
</tr>
<tr>
<td></td>
<td>• Asia Pacific (Seoul)</td>
</tr>
<tr>
<td></td>
<td>• Asia Pacific (Sydney)</td>
</tr>
<tr>
<td></td>
<td>• Europe (Frankfurt)</td>
</tr>
<tr>
<td></td>
<td>• Europe (Ireland)</td>
</tr>
<tr>
<td>Date</td>
<td>Documentation Update</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>January 24, 2019</td>
<td>The default configuration now includes a policy to purge inactive destinations.</td>
</tr>
<tr>
<td>January 17, 2019</td>
<td>Amazon MQ mq.t2.micro instance types now support only 100 connections per wire-level protocol. For more information, see Limits in Amazon MQ (p. 100).</td>
</tr>
<tr>
<td>December 19, 2018</td>
<td>You can configure a series of Amazon MQ brokers in a network of brokers. For more information, see the following sections:</td>
</tr>
<tr>
<td></td>
<td>• Amazon MQ Network of Brokers (p. 50)</td>
</tr>
<tr>
<td></td>
<td>• Creating and Configuring a Network of Brokers (p. 15)</td>
</tr>
<tr>
<td></td>
<td>• Configure Your Network of Brokers Correctly (p. 98)</td>
</tr>
<tr>
<td></td>
<td>• networkConnector (p. 75)</td>
</tr>
<tr>
<td></td>
<td>• networkConnectionStartAsync (p. 71)</td>
</tr>
<tr>
<td>December 11, 2018</td>
<td>Amazon MQ supports ActiveMQ 5.15.8, 5.15.6, and 5.15.0.</td>
</tr>
<tr>
<td></td>
<td>• Resolved bugs and improvements in ActiveMQ:</td>
</tr>
<tr>
<td></td>
<td>• ActiveMQ 5.15.8 Release Notes</td>
</tr>
<tr>
<td></td>
<td>• ActiveMQ 5.15.7 Release Notes</td>
</tr>
<tr>
<td>December 5, 2018</td>
<td>AWS supports resource tagging to help track your cost allocation. You can tag resources when creating them, or by viewing the details of that resource. For more information, see Tagging resources.</td>
</tr>
<tr>
<td>November 19, 2018</td>
<td>AWS has expanded its SOC compliance program to include Amazon MQ as an SOC compliant service.</td>
</tr>
<tr>
<td>October 15, 2018</td>
<td>• The maximum number of groups per user is 20. For more information, see Limits Related to Users (p. 101).</td>
</tr>
<tr>
<td></td>
<td>• The maximum number of connections per broker, per wire-level protocol is 1,000. For more information, see Limits Related to Brokers (p. 100).</td>
</tr>
<tr>
<td>October 2, 2018</td>
<td>AWS has expanded its HIPAA compliance program to include Amazon MQ as a HIPAA Eligible Service.</td>
</tr>
<tr>
<td>September 27, 2018</td>
<td>Amazon MQ supports ActiveMQ 5.15.6, in addition to 5.15.0. For more information, see the following:</td>
</tr>
<tr>
<td></td>
<td>• Editing Broker Engine Version, CloudWatch Logs, and Maintenance Preferences (p. 19)</td>
</tr>
<tr>
<td></td>
<td>• Resolved bugs and improvements in the ActiveMQ documentation:</td>
</tr>
<tr>
<td></td>
<td>• ActiveMQ 5.15.6 Release Notes</td>
</tr>
<tr>
<td></td>
<td>• ActiveMQ 5.15.5 Release Notes</td>
</tr>
<tr>
<td></td>
<td>• ActiveMQ 5.15.4 Release Notes</td>
</tr>
<tr>
<td></td>
<td>• ActiveMQ 5.15.3 Release Notes</td>
</tr>
<tr>
<td></td>
<td>• ActiveMQ 5.15.2 Release Notes</td>
</tr>
<tr>
<td></td>
<td>• ActiveMQ 5.15.1 Release Notes</td>
</tr>
<tr>
<td></td>
<td>• ActiveMQ Client 5.15.6</td>
</tr>
<tr>
<td>Date</td>
<td>Documentation Update</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| August 31, 2018    | • The following metrics are available:  
  - CurrentConnectionsCount  
  - TotalConsumerCount  
  - TotalProducerCount  
  For more information, see the Broker Metrics (p. 103) section.  
  • The IP address of the broker is displayed on the Details page.  
  **Note**  
  For brokers with public accessibility disabled, the internal IP address is displayed. |
| August 30, 2018    | Amazon MQ is available in the Asia Pacific (Singapore) Region in addition to the following regions:  
  - US East (Ohio)  
  - US East (N. Virginia)  
  - US West (N. California)  
  - US West (Oregon)  
  - Asia Pacific (Tokyo)  
  - Asia Pacific (Seoul)  
  - Asia Pacific (Sydney)  
  - Europe (Frankfurt)  
  - Europe (Ireland) |
| July 30, 2018      | You can configure Amazon MQ to publish general and audit logs to Amazon CloudWatch Logs. For more information, see Configuring Amazon MQ to Publish Logs to Amazon CloudWatch Logs (p. 110). |
| July 25, 2018      | Amazon MQ is available in the Asia Pacific (Tokyo) and Asia Pacific (Seoul) Regions in addition to the following regions:  
  - US East (Ohio)  
  - US East (N. Virginia)  
  - US West (N. California)  
  - US West (Oregon)  
  - Asia Pacific (Sydney)  
  - Europe (Frankfurt)  
  - Europe (Ireland) |
<p>| July 19, 2018      | You can use AWS CloudTrail to log Amazon MQ API calls. For more information, see Logging Amazon MQ API Calls Using CloudTrail (p. 107). |</p>
<table>
<thead>
<tr>
<th>Date</th>
<th>Documentation Update</th>
</tr>
</thead>
<tbody>
<tr>
<td>June 29, 2018</td>
<td>In addition to <code>mq.t2.micro</code> and <code>mq.m4.large</code>, the following broker instance types are available for regular development, testing, and production workloads that require high throughput:</td>
</tr>
<tr>
<td></td>
<td>• <code>mq.m5.large</code></td>
</tr>
<tr>
<td></td>
<td>• <code>mq.m5.xlarge</code></td>
</tr>
<tr>
<td></td>
<td>• <code>mq.m5.2xlarge</code></td>
</tr>
<tr>
<td></td>
<td>• <code>mq.m5.4xlarge</code></td>
</tr>
<tr>
<td></td>
<td>For more information, see Broker Instance Types (p. 39).</td>
</tr>
<tr>
<td>June 27, 2018</td>
<td>Amazon MQ is available in the US West (N. California) Region in addition to the following regions:</td>
</tr>
<tr>
<td></td>
<td>• US East (Ohio)</td>
</tr>
<tr>
<td></td>
<td>• US East (N. Virginia)</td>
</tr>
<tr>
<td></td>
<td>• US West (Oregon)</td>
</tr>
<tr>
<td></td>
<td>• Asia Pacific (Sydney)</td>
</tr>
<tr>
<td></td>
<td>• Europe (Frankfurt)</td>
</tr>
<tr>
<td></td>
<td>• Europe (Ireland)</td>
</tr>
<tr>
<td>June 14, 2018</td>
<td>• You can use the <code>AWS::Amazon MQ::Broker</code> AWS CloudFormation resource to perform the following actions:</td>
</tr>
<tr>
<td></td>
<td>• Create a broker.</td>
</tr>
<tr>
<td></td>
<td>• Add configuration changes or modify users for the specified broker.</td>
</tr>
<tr>
<td></td>
<td>• Return information about the specified broker.</td>
</tr>
<tr>
<td></td>
<td>• Delete the specified broker.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong></td>
</tr>
<tr>
<td></td>
<td>When you change any property of the Amazon MQ Broker <code>ConfigurationId</code> or Amazon MQ Broker User property type, the broker is rebooted immediately.</td>
</tr>
<tr>
<td></td>
<td>• You can use the <code>AWS::Amazon MQ::Configuration</code> AWS CloudFormation resource to perform the following actions:</td>
</tr>
<tr>
<td></td>
<td>• Create a configuration.</td>
</tr>
<tr>
<td></td>
<td>• Update the specified configuration.</td>
</tr>
<tr>
<td></td>
<td>• Return information about the specified configuration.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong></td>
</tr>
<tr>
<td></td>
<td>You can use AWS CloudFormation to modify—but not delete—an Amazon MQ configuration.</td>
</tr>
<tr>
<td>June 7, 2018</td>
<td>The Amazon MQ console supports German, Brazilian Portuguese, Spanish, Italian, and Traditional Chinese.</td>
</tr>
<tr>
<td>May 17, 2018</td>
<td>The limit of number of users per broker is 250. For more information, see Limits Related to Users (p. 101).</td>
</tr>
<tr>
<td>March 13, 2018</td>
<td>Creating a broker takes about 15 minutes. For more information, see Finish creating the broker (p. 13).</td>
</tr>
<tr>
<td>Date</td>
<td>Documentation Update</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>March 1, 2018</td>
<td>• You can configure the concurrent store and dispatch (p. 97) for Apache KahaDB using the <code>concurrentStoreAndDispatchQueues</code> (p. 76) attribute.</td>
</tr>
<tr>
<td></td>
<td>• The <code>CpuCreditBalance</code> CloudWatch metric (p. 103) is available for <code>mq.t2.micro</code> broker instance type.</td>
</tr>
<tr>
<td>January 10, 2018</td>
<td>The following changes affect the Amazon MQ console:</td>
</tr>
<tr>
<td></td>
<td>• In the broker list, the <code>Creation</code> column is hidden by default. To customize the page size and columns, choose <code>حلول</code>.</td>
</tr>
<tr>
<td></td>
<td>• On the <code>MyBroker</code> page, in the <code>Connections</code> section, choosing the name of your security group or <code>عرض</code> opens the EC2 console (instead of the VPC console). The EC2 console allows more intuitive configuration of inbound and outbound rules. For more information, see the updated Enable Inbound Connections (p. 6) section.</td>
</tr>
<tr>
<td>January 9, 2018</td>
<td>• The permission for REST operation ID <code>UpdateBroker</code> is listed correctly as <code>mq:UpdateBroker</code> on the IAM console.</td>
</tr>
<tr>
<td></td>
<td>• The erroneous <code>mq:DescribeEngine</code> permission is removed from the IAM console.</td>
</tr>
</tbody>
</table>
November 28, 2017

This is the initial release of Amazon MQ and the Amazon MQ Developer Guide.

- Amazon MQ is available in the following regions:
  - US East (Ohio)
  - US East (N. Virginia)
  - US West (Oregon)
  - Asia Pacific (Sydney)
  - Europe (Frankfurt)
  - Europe (Ireland)

Using the mq.t2.micro instance type is subject to CPU credits and baseline performance—with the ability to burst above the baseline level (for more information, see the CpuCreditBalance (p. 103) metric). If your application requires fixed performance, consider using an mq.m5.large instance type.

- You can create mq.m4.large and mq.t2.micro brokers.

Using the mq.t2.micro instance type is subject to CPU credits and baseline performance—with the ability to burst above the baseline level (for more information, see the CpuCreditBalance (p. 103) metric). If your application requires fixed performance, consider using an mq.m5.large instance type.

- You can use the ActiveMQ 5.15.0 broker engine.

- You can also create and manage brokers programmatically using Amazon MQ REST API and AWS SDKs.

- You can access your brokers by using any programming language that ActiveMQ supports and by enabling TLS explicitly for the following protocols:
  - AMQP
  - MQTT
  - MQTT over WebSocket
  - OpenWire
  - STOMP
  - STOMP over WebSocket

- You can connect to ActiveMQ brokers using various ActiveMQ clients. We recommend using the ActiveMQ Client. For more information, see Connecting a Java Application to Your Broker (p. 25).

- Your broker can send and receive messages of any size.

Amazon MQ Document History

The following table lists changes to the Amazon MQ Developer Guide. For Amazon MQ feature releases and improvements, see Amazon MQ Release Notes (p. 123).

<table>
<thead>
<tr>
<th>Date</th>
<th>Documentation Update</th>
</tr>
</thead>
<tbody>
<tr>
<td>December 16, 2019</td>
<td>- Added the following sections:</td>
</tr>
<tr>
<td></td>
<td>• Broker Storage (p. 44)</td>
</tr>
<tr>
<td></td>
<td>• Choose the Correct Broker Storage Type for the Best Throughput (p. 98)</td>
</tr>
<tr>
<td></td>
<td>• Revised the information in the following sections:</td>
</tr>
<tr>
<td>Date</td>
<td>Documentation Update</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>July 19, 2019</td>
<td>Modified and added content on encryption management in the following sections:&lt;br&gt;- Encryption (p. 116)&lt;br&gt;- Encryption at Rest (p. 116)&lt;br&gt;- Encryption in Transit (p. 116)</td>
</tr>
<tr>
<td>April 22, 2019</td>
<td>Added the following sections for tag-based policies and resource-level permissions:&lt;br&gt;- Tag-based Policies (p. 114)&lt;br&gt;- Resource-Level Permissions for Amazon MQ API Actions (p. 119)</td>
</tr>
<tr>
<td>March 4, 2019</td>
<td>Improved the documentation for configuring dynamic failover and the rebalancing of clients for a network of brokers. Enable dynamic failover by configuring transportConnectors along with networkConnectors configuration options.&lt;br&gt;- Dynamic Failover With Transport Connectors (p. 59)&lt;br&gt;- Amazon MQ Network of Brokers (p. 50)&lt;br&gt;- Amazon MQ Broker Configuration Parameters (p. 61)</td>
</tr>
<tr>
<td>January 5, 2019</td>
<td>Improved documentation on some per-minute metrics. For more information, see the following: Destination (Queue and Topic) Metrics (p. 106).</td>
</tr>
<tr>
<td>December 19, 2018</td>
<td>Added the following sections:&lt;br&gt;- Amazon MQ Network of Brokers (p. 50)&lt;br&gt;- Creating and Configuring a Network of Brokers (p. 15)&lt;br&gt;- Configure Your Network of Brokers Correctly (p. 98)&lt;br&gt;- networkConnector (p. 75)&lt;br&gt;- networkConnectionStartAsync (p. 71)&lt;br&gt;- Added the networkConnectors child collection element to the Elements, Child Collection Elements, and Their Child Elements Permitted in Amazon MQ Configurations (p. 71) section.</td>
</tr>
<tr>
<td>December 11, 2018</td>
<td>Updated documentation to reflect availability of ActiveMQ version 5.15.8.</td>
</tr>
<tr>
<td>December 5, 2018</td>
<td>Added the Tagging resources (p. 85) section.</td>
</tr>
<tr>
<td>October 26, 2018</td>
<td>Added the Avoid Slow Restarts by Recovering Prepared XA Transactions (p. 98) section.</td>
</tr>
<tr>
<td>October 15, 2018</td>
<td>Updated the Limits in Amazon MQ (p. 100) section.</td>
</tr>
<tr>
<td>Date</td>
<td>Documentation Update</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>October 3, 2018</td>
<td>Corrected outdated links in the Setting Up Amazon MQ (p. 3) and Amazon MQ Tutorials (p. 11) sections.</td>
</tr>
<tr>
<td>October 1, 2018</td>
<td>Corrected the information in the Next Steps (p. 10) section.</td>
</tr>
<tr>
<td>September 27, 2018</td>
<td>• Added the Editing Broker Engine Version, CloudWatch Logs, and Maintenance Preferences (p. 19) section.</td>
</tr>
<tr>
<td></td>
<td>• Updated the following sections:</td>
</tr>
<tr>
<td></td>
<td>• Broker Engine (p. 43)</td>
</tr>
<tr>
<td></td>
<td>• Create an ActiveMQ Broker (p. 5)</td>
</tr>
<tr>
<td></td>
<td>• Configure Basic Broker Settings (p. 11)</td>
</tr>
<tr>
<td>September 18, 2018</td>
<td>Added the following note to the Creating and Managing Amazon MQ Broker Users (p. 30) section: You can't configure groups independently of users. A group label is created when you add at least one user to it and deleted when you remove all users from it.</td>
</tr>
<tr>
<td>August 31, 2018</td>
<td>• Clarified the terminology for active/standby brokers. For more information, see Amazon MQ Active/Standby Broker for High Availability (p. 49).</td>
</tr>
<tr>
<td></td>
<td>• Simplified the terminology for the maintenance window. For more information, see Amazon MQ Broker Configuration Lifecycle (p. 60).</td>
</tr>
<tr>
<td></td>
<td>• Rewrote the Configure Additional Broker Settings (p. 12) section.</td>
</tr>
<tr>
<td></td>
<td>• Updated the Broker Metrics (p. 103) and Listing Brokers and Viewing Broker Details (p. 29) sections.</td>
</tr>
<tr>
<td>August 15, 2018</td>
<td>Corrected the information in the Create an ActiveMQ Broker (p. 5) section.</td>
</tr>
<tr>
<td>August 13, 2018</td>
<td>Added the Accessing the ActiveMQ Web Console of a Broker without Public Accessibility (p. 14) section.</td>
</tr>
<tr>
<td>August 2, 2018</td>
<td>• Added the Troubleshooting CloudWatch Logs Configuration (p. 112) section.</td>
</tr>
<tr>
<td></td>
<td>• Added the following admonition throughout this guide:</td>
</tr>
<tr>
<td></td>
<td><strong>Important</strong></td>
</tr>
<tr>
<td></td>
<td>In the following example code, producers and consumers run in a single thread. For production systems (or to test broker instance failover), make sure that your producers and consumers run on separate hosts or threads.</td>
</tr>
<tr>
<td>August 1, 2018</td>
<td>Corrected the information in the following sections:</td>
</tr>
<tr>
<td></td>
<td>• Understanding the Structure of Logging in CloudWatch Logs (p. 111)</td>
</tr>
<tr>
<td></td>
<td>• Connect a Java Application to Your Broker (p. 6)</td>
</tr>
<tr>
<td>July 31, 2018</td>
<td>• Moved the 3-minute demo video to the Getting Started with Amazon MQ (p. 5) section.</td>
</tr>
<tr>
<td></td>
<td>• Added the 3-minute getting started video to the What is Amazon MQ? (p. 1) section.</td>
</tr>
<tr>
<td>July 30, 2018</td>
<td>• Added the Configuring Amazon MQ to Publish Logs to Amazon CloudWatch Logs (p. 110) section.</td>
</tr>
<tr>
<td></td>
<td>• Updated the Configure Additional Broker Settings (p. 12) section.</td>
</tr>
<tr>
<td>Date</td>
<td>Documentation Update</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| July 19, 2018    | • Added the Logging Amazon MQ API Calls Using CloudTrail (p. 107) section.  
• Corrected the information in the What Are the Main Benefits of Amazon MQ? (p. 1) section.                                                                 |
| July 5, 2018     | • Added an authorizationEntry child element cross-reference to the Always Configure an Authorization Map (p. 94) section.  
• Clarified the information in the Messaging Authentication and Authorization for ActiveMQ (p. 120) section.  
• Clarified the information in the Limits Related to API Throttling (p. 102) section.                                                                 |
| June 29, 2018    | • Updated the information in the Broker Instance Types (p. 39) section.  
• Added the Choose the Correct Broker Instance Type for the Best Throughput (p. 97) section.                                                                 |
| June 26, 2018    | Added a link to a related resource to the Migrating to Amazon MQ (p. 88) section.                                                                                                                               |
| June 4, 2018     | In addition to GitHub, HTML, PDF, and Kindle, the Amazon MQ Developer Guide release notes are available as an RSS feed.                                                                                         |
| May 29, 2018     | Made the following changes in the Working Java Example (p. 78) section:  
• Added a STOMP+WSS Java example. The STOMP+WSS example Java code connects to a broker, creates a queue, and publishes and receives a message.  
• Improved the MQTT Java example.  
• Improved the OpenWire Java example.                                                                                                                      |
| May 24, 2018     | Corrected the wire-level protocol endpoint port in the MQTT Java example in the Working Java Example (p. 78) section.                                                                                             |
| May 22, 2018     | Corrected the information in all Java dependency sections.                                                                                                                                                        |
| May 17, 2018     | Corrected the information in the Limits Related to Users (p. 101) section.                                                                                                                                       |
| May 15, 2018     | Corrected the information in the Ensuring Effective Amazon MQ Performance (p. 96) section.                                                                                                                        |
| May 8, 2018      | • Placed the Amazon MQ REST API Permissions Reference (p. 118) in its own section.  
• Created the IAM Permissions Required to Create an Amazon MQ Broker (p. 117) section with an example custom IAM policy.                                                                                       |
| May 7, 2018      | • Clarified throughout this guide that the broker maintenance window is 2 hours long. For more information, see Amazon MQ Broker Configuration Lifecycle (p. 60).  
• Added explanations for why the ec2:CreateNetworkInterface and ec2:CreateNetworkInterfacePermission permissions are necessary for creating a broker. For more information, see API Authentication and Authorization for Amazon MQ (p. 117). |
### Amazon MQ Developer Guide

#### Document History

<table>
<thead>
<tr>
<th>Date</th>
<th>Documentation Update</th>
</tr>
</thead>
<tbody>
<tr>
<td>May 1, 2018</td>
<td>Clarified the information about the maintenance window for active/standby brokers in the following sections:</td>
</tr>
<tr>
<td></td>
<td>- Amazon MQ Active/Standby Broker for High Availability (p. 49)</td>
</tr>
<tr>
<td></td>
<td>- Creating and Configuring a Broker (p. 11)</td>
</tr>
<tr>
<td></td>
<td>- Creating and Applying Broker Configurations (p. 20)</td>
</tr>
<tr>
<td></td>
<td>- Editing and Managing Broker Configurations (p. 22)</td>
</tr>
<tr>
<td>April 27, 2018</td>
<td>Rewrote the following sections and optimized example Java code to match the recommendation to use connection pooling only for producers, not consumers:</td>
</tr>
<tr>
<td></td>
<td>- Always Use Connection Pooling (p. 95)</td>
</tr>
<tr>
<td></td>
<td>- Create a Message Producer and Send a Message (p. 7)</td>
</tr>
<tr>
<td></td>
<td>- Create a Message Consumer and Receive the Message (p. 8)</td>
</tr>
<tr>
<td></td>
<td>- AmazonMQExample.java (p. 80)</td>
</tr>
<tr>
<td>April 26, 2018</td>
<td>Added an MQTT Java example to the Working Java Example (p. 78) section. The MQTT example Java code connects to a broker, creates a topic, and publishes and receives a message.</td>
</tr>
<tr>
<td>April 4, 2018</td>
<td>Renamed the Communicating with Amazon MQ section to Connecting to Amazon MQ (p. 94).</td>
</tr>
<tr>
<td>April 3, 2018</td>
<td>Clarified and corrected the information in the Disable Concurrent Store and Dispatch for Queues with Slow Consumers (p. 97) section.</td>
</tr>
<tr>
<td>April 2, 2018</td>
<td>Moved the Concurrent Store and Dispatch for Queues in Amazon MQ section to the Disable Concurrent Store and Dispatch for Queues with Slow Consumers (p. 97) section.</td>
</tr>
<tr>
<td>March 27, 2018</td>
<td>- Replaced the re:Invent launch video with a 3-minute demo video in the What is Amazon MQ? (p. 1) section.</td>
</tr>
<tr>
<td></td>
<td>- Rewrote the What Are the Main Benefits of Amazon MQ? (p. 1) section to include information about recently released features.</td>
</tr>
<tr>
<td></td>
<td>- Restructured the following sections:</td>
</tr>
<tr>
<td></td>
<td>- Amazon MQ Broker Architecture (p. 45)</td>
</tr>
<tr>
<td></td>
<td>- How Amazon MQ Works (p. 37)</td>
</tr>
<tr>
<td></td>
<td>- Migrating to Amazon MQ (p. 88)</td>
</tr>
<tr>
<td></td>
<td>- Moved Amazon MQ Broker Configuration Lifecycle (p. 60) under the Amazon MQ Broker Architecture (p. 45) section.</td>
</tr>
<tr>
<td>March 22, 2018</td>
<td>Clarified the following statement throughout this guide: Amazon MQ encrypts messages at rest and in transit using encryption keys that it manages and stores securely. For additional security, we highly recommend designing your application to use client-side encryption. For more information, see the AWS Encryption SDK Developer Guide.</td>
</tr>
<tr>
<td>March 19, 2018</td>
<td>Clarified the following statement throughout this guide: An Active/standby broker is comprised of two brokers in two different Availability Zones, configured in a redundant pair. These brokers communicate synchronously with your application, and with Amazon EFS.</td>
</tr>
<tr>
<td>Date</td>
<td>Documentation Update</td>
</tr>
<tr>
<td>------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| March 15, 2018   | • Restructured the Amazon MQ Basic Elements (p. 37) section.  
• Improved the explanation of the diagrams in the following sections:  
  • Migrating to Amazon MQ without Service Interruption (p. 88)  
  • Migrating to Amazon MQ with Service Interruption (p. 90) |
| March 12, 2018   | • Clarified and corrected the information in the Using Amazon MQ Securely (p. 93) and Connecting to Amazon MQ (p. 94) sections.  
• Added the Disable Concurrent Store and Dispatch for Queues with Slow Consumers (p. 97) section.  
• Grouped admonitions into a preface for the Configure advanced broker settings (p. 12) section. |
| March 9, 2018    | • Clarified and corrected the information in the Always Configure an Authorization Map (p. 94) section.  
• Added the authorizationEntry (p. 74) section and updated the kahaDB (p. 76) section.                                      |
| March 8, 2018    | • Added the Always Configure an Authorization Map (p. 94) section.  
• Added notes about broker suffixes to the Monitoring Amazon MQ Using CloudWatch (p. 103) section.                                                            |
| March 6, 2018    | Added the following note throughout this guide:  
**Note**  
Using the `mq.t2.micro` instance type is subject to CPU credits and baseline performance—with the ability to burst above the baseline level (for more information, see the `CpuCreditBalance` (p. 103) metric). If your application requires fixed performance, consider using an `mq.m5.large` instance type. |
| March 1, 2018    | • Added the `CpuCreditBalance` metric to the Broker Metrics (p. 103) section.  
• Added the Amazon MQ Child Element Attributes (p. 74) section.  
• Added links from elements in the section called “Permitted Elements” (p. 62) section to their attributes and to child collection elements.  
• Made corrections to the AWS Glossary in GitHub. |
| February 28, 2018| Corrected image display in GitHub.                                                                                                                                                                                  |
| February 27, 2018| In addition to HTML, PDF, and Kindle, the Amazon MQ Developer Guide is available on GitHub. To leave feedback, choose the GitHub icon in the upper right-hand corner.                                                  |
| February 26, 2018| • Made regions consistent in all examples and diagrams.  
• Optimized links to the AWS console and product webpages.                                                                                                   |
<table>
<thead>
<tr>
<th>Date</th>
<th>Documentation Update</th>
</tr>
</thead>
<tbody>
<tr>
<td>February 22, 2018</td>
<td>Clarified and corrected the information in the following sections:</td>
</tr>
<tr>
<td></td>
<td>• Prefer Brokers without Public Accessibility (p. 93)</td>
</tr>
<tr>
<td></td>
<td>• Always Use Client-Side Encryption as a Complement to TLS (p. 93)</td>
</tr>
<tr>
<td></td>
<td>• Always Use the Failover Transport to Connect to Multiple Broker Endpoints (p. 96)</td>
</tr>
<tr>
<td></td>
<td>• API Authentication and Authorization for Amazon MQ (p. 117)</td>
</tr>
<tr>
<td></td>
<td>• Messaging Authentication and Authorization for ActiveMQ (p. 120)</td>
</tr>
<tr>
<td>February 21, 2018</td>
<td>Corrected the Java code in the following sections:</td>
</tr>
<tr>
<td></td>
<td>• Working Java Example (p. 78)</td>
</tr>
<tr>
<td></td>
<td>• Connect a Java Application to Your Broker (p. 6)</td>
</tr>
<tr>
<td></td>
<td>• Always Use Connection Pooling (p. 95)</td>
</tr>
<tr>
<td>February 20, 2018</td>
<td>Clarified and corrected the information in the Amazon MQ Security (p. 114) and Best Practices for Amazon MQ (p. 93) sections.</td>
</tr>
<tr>
<td>February 19, 2018</td>
<td>• Corrected the Java code in the Always Use Connection Pooling (p. 95) section.</td>
</tr>
<tr>
<td></td>
<td>• Clarified and corrected the information in the Always Use Client-Side Encryption as a Complement to TLS (p. 93) section.</td>
</tr>
<tr>
<td></td>
<td>• Restructured and expanded the Best Practices for Amazon MQ (p. 93) and Amazon MQ Security (p. 114) sections.</td>
</tr>
<tr>
<td>February 16, 2018</td>
<td>• Added the Using Amazon MQ Securely (p. 93) section.</td>
</tr>
<tr>
<td></td>
<td>• Updated the Connecting to Amazon MQ (p. 94) section.</td>
</tr>
<tr>
<td></td>
<td>• Corrected the Java code in the following sections:</td>
</tr>
<tr>
<td></td>
<td>• Getting Started with Amazon MQ (p. 5)</td>
</tr>
<tr>
<td></td>
<td>• AmazonMQExample.java (p. 80)</td>
</tr>
<tr>
<td>February 15, 2018</td>
<td>• Restructured and expanded the Best Practices for Amazon MQ (p. 93) section.</td>
</tr>
<tr>
<td></td>
<td>• Updated the following sections:</td>
</tr>
<tr>
<td></td>
<td>• How Can I Get Started with Amazon MQ? (p. 2)</td>
</tr>
<tr>
<td></td>
<td>• Next Steps (p. 10) (Getting Started)</td>
</tr>
<tr>
<td></td>
<td>• Related Resources (p. 121)</td>
</tr>
<tr>
<td>February 14, 2018</td>
<td>Updated the following sections:</td>
</tr>
<tr>
<td></td>
<td>• Limits in Amazon MQ (p. 100)</td>
</tr>
<tr>
<td></td>
<td>• Limits Related to API Throttling (p. 102)</td>
</tr>
<tr>
<td></td>
<td>• Best Practices for Amazon MQ (p. 93)</td>
</tr>
<tr>
<td></td>
<td>• Amazon MQ Security (p. 114)</td>
</tr>
<tr>
<td>February 13, 2018</td>
<td>• Updated the Related Resources (p. 121) section.</td>
</tr>
<tr>
<td></td>
<td>• Updated the Limits in Amazon MQ (p. 100) section.</td>
</tr>
<tr>
<td></td>
<td>• Added the We Want to Hear from You (p. 2) section.</td>
</tr>
<tr>
<td>January 25, 2018</td>
<td>• Fixed an error in the Add Java Dependencies (p. 79) subsection of the Working Java Example (p. 78) section.</td>
</tr>
<tr>
<td></td>
<td>• The permission for REST operation ID RebootBroker is listed correctly as mq:RebootBroker on the IAM console.</td>
</tr>
<tr>
<td>Date</td>
<td>Documentation Update</td>
</tr>
<tr>
<td>--------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| January 24, 2018   | • Added the Never Modify or Delete the Amazon MQ Elastic Network Interface (p. 94) section.  
• Updated all diagrams throughout this guide.  
• Added links to the Amazon MQ REST API Reference throughout this guide and links to specific REST APIs to the API Authentication and Authorization for Amazon MQ (p. 117) section. |
| January 19, 2018   | Updated the information in the Apache ActiveMQ Resources (p. 121) section.                                                                                                                                                                                                                                                                             |
| January 18, 2018   | Clarified and corrected the information in the Limits in Amazon MQ (p. 100) section.                                                                                                                                                                                                                                                                     |
| January 17, 2018   | Reinstated the recommendation to prefer virtual destinations over durable subscriptions (p. 96), with an improved explanation.                                                                                                                                                                                                                     |
| January 11, 2018   | • The Amazon MQ Developer Guide is available in Kindle format, in addition to HTML and PDF.  
• Clarified and corrected information in the API Authentication and Authorization for AmazonMQ (p. 117) and Create an IAM User and Get Your AWS Credentials (p. 3) sections. |
| January 3, 2018    | Added DescribeConfigurationRevision to the API Authentication and Authorization for Amazon MQ (p. 117) section.                                                                                                                                                                                                                               |
| December 15, 2017  | Removed the recommendation against durable subscriptions from the Best Practices for Amazon MQ (p. 93) section.                                                                                                                                                                                                                                       |
| December 8, 2017   | • Added the Enable Inbound Connections (p. 6) prerequisite to the Connecting a Java Application to Your Broker (p. 25) and Working Java Example (p. 78) sections.  
• Added the following note throughout this guide: Currently, you can't delete a configuration.                                                                                                                                                                                     |
| December 7, 2017   | • Improved the code in the AmazonMQExample.java (p. 80).  
• Added the API Authentication and Authorization for Amazon MQ (p. 117) section.                                                                                                                                                                                                              |
| December 5, 2017   | • Clarified and corrected information in the Monitoring Amazon MQ Using CloudWatch (p. 103) section:  
• Improved the metric descriptions.  
• Added the Dimension for Broker Metrics (p. 105) and Dimensions for Destination (Queue and Topic) Metrics (p. 107) sub-sections.  
• Added the "Introducing Amazon MQ" video to the What is Amazon MQ? (p. 1) section.                                                                                                                                   |
| December 4, 2017   | • Clarified the following information in the Limits Related to Data Storage (p. 101) section: Storage capacity per broker is 200 GB.  
• Added the Prerequisites (p. 78) to the Working Java Example (p. 78) section. (The activemq-client.jar and activemq-pool.jar packages are required for the example to work. For more information, see Connecting a Java Application to Your Broker (p. 25)). |
<table>
<thead>
<tr>
<th>Date</th>
<th>Documentation Update</th>
</tr>
</thead>
</table>
| December 1, 2017 | • Updated and improved the screenshots in all the tutorials.  
• Clarified the following explanation throughout this guide: Making changes to a configuration revision or an ActiveMQ user does not apply the changes immediately. To apply your changes, you must wait for the next maintenance window (p. 24) or reboot the broker (p. 32). For more information, see Amazon MQ Broker Configuration Lifecycle (p. 60). |
AWS glossary

For the latest AWS terminology, see the AWS glossary in the AWS General Reference.