Red Hat AMQ 2021.Q2

Red Hat AMQ 7 の概要

機能およびコンポーネントの概要
機能およびコンポーネントの概要

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.
法律上の通知

Copyright © 2021 | You need to change the HOLDER entity in the en-US/Introducing_Red_Hat_AMQ_7.ent file |

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at http://creativecommons.org/licenses/by-sa/3.0/ . In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux® is the registered trademark of Linus Torvalds in the United States and other countries.

Java® is a registered trademark of Oracle and/or its affiliates.

XFS® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js® is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

概要

本書では、Red Hat AMQ 7 の機能およびコンポーネントについて詳しく説明します。また、本リリースでサポートされる一般的なユースケースと設計パターンも示しています。
第1章 RED HAT AMQ について

1.1. 主な特長

インターネット規模でのメッセージング
最上級のセキュリティとパフォーマンス
幅広いプラットフォームおよび言語のサポート
標準化に焦点
管理の一元化

第2章 コンポーネントの概要

2.1. AMQ BROKER
2.2. AMQ INTERCONNECT
2.3. AMQ CLIENTS

AMQP クライアント
JMS クライアント
アダプターおよびライブラリー

2.4. コンポーネントの互換性

第3章 一般的なデプロイメントパターン

3.1. 中央ブローカー
3.2. ルーティング対応メッセージング
3.3. 高可用性ブローカー
3.4. ロードバランサーの背後のルーターペア
3.5. DMZ のルーターペア
3.6. 異なるデータセンターのルーターペア
第1章 RED HAT AMQ 7 について

Red Hat AMQ は、インターネット向けアプリケーションに対して高速で軽量でセキュアなメッセージングを提供します。AMQ Broker は複数のプロトコルと高速メッセージの永続性をサポートします。AMQ Interconnect は AMQP プロトコルを活用して、ネットワーク全体でメッセージングリソースを配置およびスケーリングします。AMQ Clients は、複数の言語およびプラットフォームに対してメッセージング API のスイートを提供します。

AMQ コンポーネントをツールボックス内のツールと考えてください。メッセージングアプリケーションをビルドおよび維持するために、複数のコンポーネントを一緒に使用したり、個別に使用したりできます。AMQP はコンポーネントを一緒にバインドするツールボックス内の接着剤です。AMQ コンポーネントは共通の管理コンソールを共有するため、単一のインターフェースから管理できます。

注記

Red Hat AMQ 7 には、OpenShift Container Platform 上で Apache Kafka を実行するためのコンテナイメージおよび Operator を提供する AMQ Streams が含まれます。AMQ Streams では AMQP はサポートされません。

1.1. 主な特長

AMQ を使用すると、開発者は高速で信頼性があり、管理しやすいメッセージングアプリケーションビルドできます。

インターネット規模でのメッセージング
AMQ には、高度なマルチデータセンターメッセージングネットワークを構築するためのツールが含まれています。クライアント、ブローカー、およびスタンドアロンのサービスをシームレスなメッセージングファブリックで接続できます。

最上部のセキュリティーとパフォーマンス
AMQ は最新の SSL/TLS 暗号化、および拡張可能な SASL 認証を提供します。AMQ は、高速で高ボリュームのメッセージングとクラスをリードする JMS パフォーマンスを提供します。

幅広いプラットフォームおよび言語のサポート
AMQ は複数の言語やオペレーティングシステムに対応するため、多様なアプリケーションコンポーネントが通信できます。AMQ は、C++、Java、JavaScript、Python、Ruby、および .NET アプリケーション、Linux、Windows、および JVM ベースの環境をサポートします。

標準化に焦点
AMQ は Java JMS 1.1 および 2.0 API 仕様を実装します。そのコンポーネントは、ISO 規格の AMQP 1.0 および MQTT メッセージングプロトコル、ならびに STOMP および WebSocket をサポートします。

管理の一元化
AMQ では、単一の管理インタフェースからすべての AMQ コンポーネントを管理できます。JMX または REST インタフェースを使用して、プログラム的にサーバーを管理できます。
第2章 コンポーネントの概要

Red Hat AMQ は AMQ Broker、AMQ Interconnect、および AMQ Clients で構成されます。これらは連携して、分散アプリケーションでのネットワーク通信を有効にします。

- AMQ Broker
- AMQ Interconnect
- AMQ Clients

2.1. AMQ BROKER

AMQ Broker はフル機能のメッセージ指向ミドルウェアブローカーです。これにより、高度なアドレスングとキューリング、高速メッセージの永続性、および高可用性を提供します。AMQ Broker は複数のプロトコルとオペレーティング環境をサポートしているので、既存のアセットを使用できます。AMQ Broker は Red Hat JBoss Enterprise Application Platform とのインテグレーションをサポートします。

詳細は、『Getting Started with AMQ Broker』を参照してください。

2.2. AMQ INTERCONNECT

AMQ Interconnect は、クライアント、ブローカー、スタンドアロンサービスなど、AMQP 対応のエンポイント間のメッセージを柔軟にルーティングします。AMQ Interconnect ルーターのネットワークへの单一の接続を使用して、クライアントはネットワークに接続された他のエンドポイントとのメッセージを交換できます。

AMQ Interconnect では、高可用性のためにマスター/スレーブクラスターを使用しません。これは通常、冗長ネットワークパスを持つ複数ルーターのトポロジーにデプロイされ、これを使用して信頼できる接続を提供します。AMQ Interconnect は、メッセージングワークロードをネットワーク全体に分散し、低レイテンシーで新たなレベルのスケーリングを実現できます。

詳細は、『Using the AMQ Interconnect router』および『Release Notes for AMQ Interconnect 2.0』を参照してください。

2.3. AMQ CLIENTS

AMQ Clients は AMQP 1.0 および JMS クライアント、アダプター、およびライブラリーのスイートです。これには、JMS 2.0 のサポートおよび既存のアプリケーションへのインテグレーションを可能にする新しいイベント駆動型 API が含まれます。

詳細は、『AMQ Clients Overview』を参照してください。

AMQP クライアント
- AMQ C++
- AMQ JavaScript
- AMQ JMS (Java)
- AMQ .NET
- AMQ Python
JMS クライアント

- AMQ JMS (AMQP 1.0)
- AMQ Core Protocol JMS
- AMQ OpenWire JMS

アダプターおよびライブラリー

- AMQ JMS Pool
- AMQ Spring Boot Starter

2.4. コンポーネントの互換性

以下の表は、AMQ コンポーネントのサポートされている言語、プラットフォーム、およびプロトコルを示しています。同じプロトコルをサポートするコンポーネントは、言語やプラットフォームが異なる場合でも相互運用できることに注意してください。たとえば、AMQ Python は AMQ JMS と通信できます。

表2.1 AMQ コンポーネントの互換性

<table>
<thead>
<tr>
<th>コンポーネント</th>
<th>言語</th>
<th>プラットフォーム</th>
<th>プロトコル</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMQ Broker</td>
<td>-</td>
<td>JVM</td>
<td>AMQP 1.0、MQTT、OpenWire、STOMP、Core Protocol</td>
</tr>
<tr>
<td>AMQ Interconnect</td>
<td>-</td>
<td>Linux</td>
<td>AMQP 1.0</td>
</tr>
<tr>
<td>AMQ C++</td>
<td>C++</td>
<td>Linux、Windows</td>
<td>AMQP 1.0</td>
</tr>
<tr>
<td>AMQ JavaScript</td>
<td>JavaScript</td>
<td>Node.js、ブラウザ</td>
<td>AMQP 1.0</td>
</tr>
<tr>
<td>AMQ JMS</td>
<td>Java</td>
<td>JVM</td>
<td>AMQP 1.0</td>
</tr>
<tr>
<td>AMQ .NET</td>
<td>C#</td>
<td>.NET</td>
<td>AMQP 1.0</td>
</tr>
<tr>
<td>AMQ Python</td>
<td>Python</td>
<td>Linux</td>
<td>AMQP 1.0</td>
</tr>
<tr>
<td>AMQ Ruby</td>
<td>Ruby</td>
<td>Linux</td>
<td>AMQP 1.0</td>
</tr>
<tr>
<td>AMQ Spring Boot Starter</td>
<td>Java</td>
<td>JVM</td>
<td>AMQP 1.0</td>
</tr>
<tr>
<td>AMQ Core Protocol JMS</td>
<td>Java</td>
<td>JVM</td>
<td>Core Protocol</td>
</tr>
<tr>
<td>AMQ OpenWire JMS</td>
<td>Java</td>
<td>JVM</td>
<td>OpenWire</td>
</tr>
<tr>
<td>コンポーネント</td>
<td>言語</td>
<td>プラットフォーム</td>
<td>プロトコル</td>
</tr>
<tr>
<td>-----------------</td>
<td>------</td>
<td>----------------</td>
<td>------------</td>
</tr>
<tr>
<td>AMQ JMS Pool</td>
<td>Java</td>
<td>JVM</td>
<td>-</td>
</tr>
</tbody>
</table>

詳細は、「Red Hat AMQ 7 Supported Configurations」を参照してください。
第3章 一般的なデプロイメントパターン

Red Hat AMQ 7 は、さまざまなトポロジーで設定できます。以下は、AMQ コンポーネントを使用して実装できる一般的なデプロイメントパターンの一部です。

3.1. 中央ブローカー

中央ブローカーパターンは、比較的簡単に設定でき、維持することができます。また、これは比較的堅牢です。ルートは通常ローカルです。これは、追加されたノード数に関係なく、ブローカーとそのクラウドは常に互いに1ネットワークホップ内にあるためです。このパターンはハブとスポークとも呼ばれ、中央ブローカーがハブに、クライアントがスポークに、それぞれ相当します。

図3.1 中央ブローカーパターン

重要な要素は中央のブローカーノードだけです。メンテナンスの作業のフォーカスとして、このブローカーをクライアントで利用可能にすることになります。

3.2. ルーティング対応メッセージング

メッセージをリモートの宛先にルーティングする場合、ブローカーはそれらをローカルキューニーに保存してから宛先に転送します。ただし、アプリケーションがリクエストおよび応答メッセージをリアルタイムで送信することを要求し、ブローカーストアと転送メッセージを持つことで大きなコストがかかる場合があります。AMQ では、ブローカーの代わりにルーターを使用して、このようなコストを回避することができます。ブローカーとは異なり、ルーターはメッセージを宛先に転送する前に保存しません。その代わりに、ライトウェイトなパイプとして動作し、2つのエンドポイントを直接接続します。
図3.2 ブローカーレスルーティング対応メッセージングパターン

図3.3 高可用性ブローカー

クライアントがブローカーを使用できるようにするには、高可用性(HA)のマスター/スレーブペアをデプロイして、バックアップグループを作成します。たとえば、2つのノードに2つのマスター/スレーブグループをデプロイできます。このようなデプロイメントは、以下の図のように、アクティブな各ブローカーのバックアップを提供します。

図3.3 マスター/スレーブペア

通常の運用条件では1つのマスターブローカーが各ノード(物理サーバーまたは仮想マシンのいずれか)でアクティブになります。1つのノードに障害が発生すると、他のノード上のスレーブが引き継ぎます。結果として、同じ正常なノードに2つのアクティブなブローカーが存在します。

マスター/スレーブペアをデプロイすることで、このようなバックアップグループのネットワーク全体をスケールアウトすることができます。このタイプの大規模なデプロイメントは、メッセージ処理の負荷を多くのブローカーに分散する場合に便利です。以下の図のブローカーネットワークは、8つのノードに分散される8つのマスター/スレーブグループで構成されます。
3.4. ロードバランサーの背後のルーターペア

ロードバランサーの背後に2つのルーターをデプロイすると、単一データセンターのデプロイメントの可用性、回復性、およびスケーラビリティーが向上します。エンドポイントは、ロードバランサーによってサポートされる既知のURLへの接続を作成します。次に、ロードバランサーはルーター間で受信接続を分散し、接続とメッセージングの負荷が分散されるようにします。ルーターの1つが失敗すると、そのルーターに接続されているエンドポイントは残りのアクティブなルーターに再接続します。
スケーラビリティーをさらに向上させるために、多くのルーター（たとえば3つまたは4つ）を使用することができます。各ルーターは、他のすべてのルーターに直接接続します。

### 3.5. DMZのルーターペア

このデプロイメントアーキテクチャーでは、ルーターネットワークは、外部のクライアントとエンタープライズアプリケーションをホストするブローカーとの間の保護と分離のレイヤーを提供します。

![DMZのルーターペアの図](image)

#### DMZトポロジーに関する重要事項

- デプロイメント内の接続のセキュリティーは、外部クライアントに使用されるセキュリティーとは異なります。たとえば、お使いのデプロイメントでは、内部のセキュリティー用にプライベート認証局 (CA) を使用して認証用に各ルーターおよびブローカーに x.509 証明書を発行する一方、外部のユーザーは別のパブリック CA を使用する可能性があります。

- エンタープライズと DMZとの間のルーター間接続は常に、セキュリティーを確保するためエンタープライズから DMZに確立されます。したがって、外部からエンタープライズへの接続は許可されません。ただし、AMQP プロトコルは、接続を確立した後に双方向通信を有効にします。

### 3.6. 異なるデータセンターのルーターペア

AMQ コンポーネントのデプロイメントで、複数のロケーションにまたがるより複雑なトポロジーを使用できます。たとえば、負荷分散されたルーターのペアを、4つのロケーションのそれぞれにデプロイすることが可能です。センターに2つのバックボーンのルーターを追加して、全ロケーション間の冗長な接続を提供する場合があります。以下の図は、複数のロケーションにまたがるデプロイメントの例です。
図3.7 複数の相互接続されたルーター

<table>
<thead>
<tr>
<th>LOCATION 1</th>
<th>LOCATION 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Broker</td>
<td>Broker</td>
</tr>
<tr>
<td>Broker</td>
<td>Broker</td>
</tr>
<tr>
<td>Broker</td>
<td>Broker</td>
</tr>
<tr>
<td>Router</td>
<td>Router</td>
</tr>
<tr>
<td>Load Balancer</td>
<td>Load Balancer</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LOCATION 3</th>
<th>LOCATION 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Broker</td>
<td>Broker</td>
</tr>
<tr>
<td>Broker</td>
<td>Broker</td>
</tr>
<tr>
<td>Broker</td>
<td>Broker</td>
</tr>
<tr>
<td>Router</td>
<td>Router</td>
</tr>
<tr>
<td>Load Balancer</td>
<td>Load Balancer</td>
</tr>
</tbody>
</table>

Revised on 2021-08-01 15:37:31 +1000