Red Hat AMQ 2021.Q1

Red Hat AMQ 7 の概要

機能およびコンポーネントの概要

Last Updated: 2023-02-27

Red Hat AMQ 2021.Q1 Red Hat AMQ 7 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

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。

目次

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信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。

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をビルドできます。

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、単一の管理インター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 インター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、AMQ Clients で構成されます。これは連携して、分散アプリケーションでのネットワーク通信を可能にします。

Red Hat AMQ には、Apache Kafka をベースとした AMQ Streams も含まれています。AMQ Streams は AMQP をサポートしないか、Red Hat AMQ Console を使用します。

- AMQ Broker
- AMQ Interconnect
- AMQ Clients
- AMQ Streams

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、高度なアドレシングとキューイング、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 対応のエンド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単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、「AMQ Interconnect の使用」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 の概要」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
- AMQ Ruby

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. AMQ STREAMS

AMQ Streams は、Apache Kafka をベースとしたスケーラビリティーが極めて高く、分散型の高パフォーマンスのデータストリーミングプラットフォームです。AMQ Streams は、OpenShift クラスターで Apache Kafka を実行するプロセスを簡素化します。Red Hat Enterprise Linux にインストールすることもできます。

AMQ Streams は、Kafka を OpenShift で実行するためのコンテナーイメージおよび Operator を提供します。AMQ Streams Operator は、AMQ Streams の実行に必要です。AMQ Streams で提供される Operator は、Kafka を効果的に管理するために、専門的なオペレーション情報で目的に合うよう構築されています。

Cluster Operator

Apache Kafka クラスター、Kafka Connect、Kafka MirrorMaker、Kafka Bridge、Kafka Exporter、および Entity Operator をデプロイおよび管理します。

Entitiy Operator

Topic Operator および User Operator を構成します。

Topic Operator

Kafka トピックを管理します。

User Operator

Kafka ブローカーにアクセスするクライアントアプリケーションを表す Kafka ユーザーを管理します。

詳細は、「 AMQ Streams on OpenShift の概要 」を参照してください。

2.5. 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

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

詳細は、「Red Hat AMQ 7 Supported Configurations」 を参照してください。

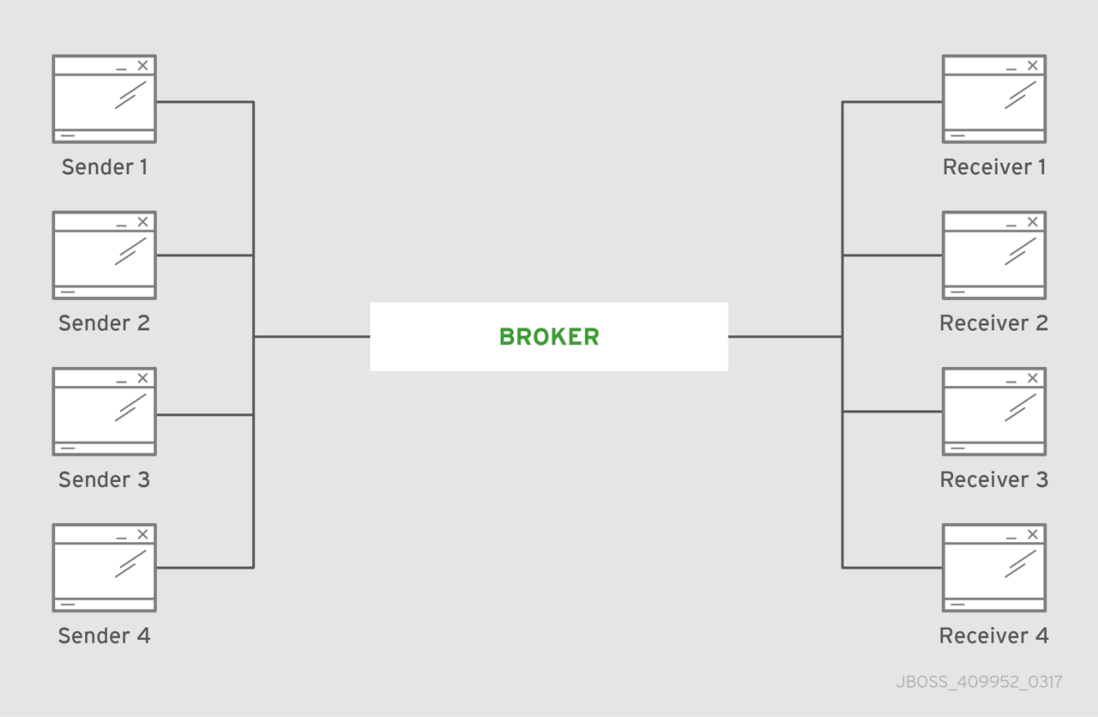
第3章 一般的なデプロイメントパターン

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イアントは常に互いに 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

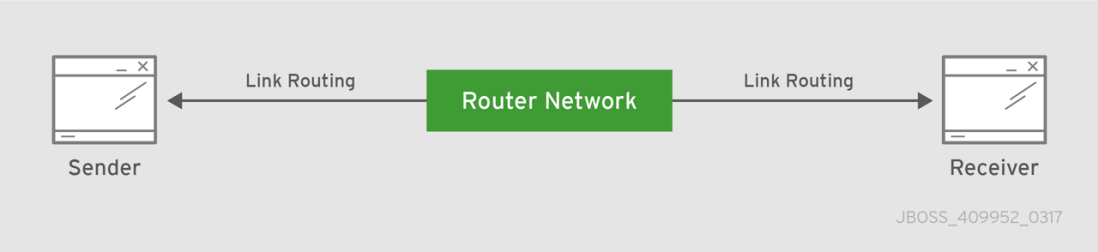


重要な要素は中央のブローカーノードだけです。メンテナンスの作業のフォーカスとして、このブローカーをクライアントで利用可能にすることになります。

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。

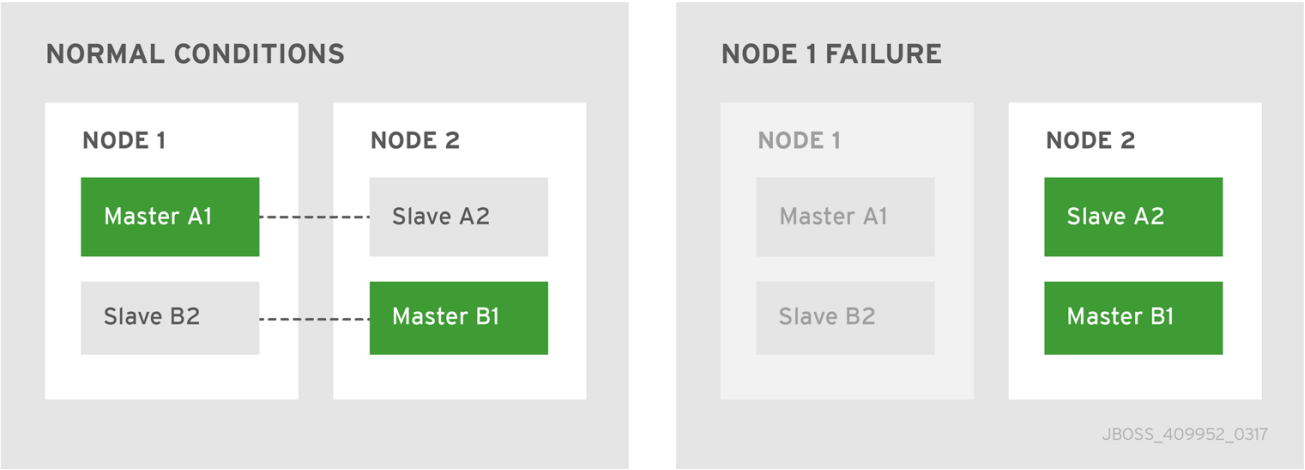
図3.2 ブローカーレスルーティング対応メッセージングパターン



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。

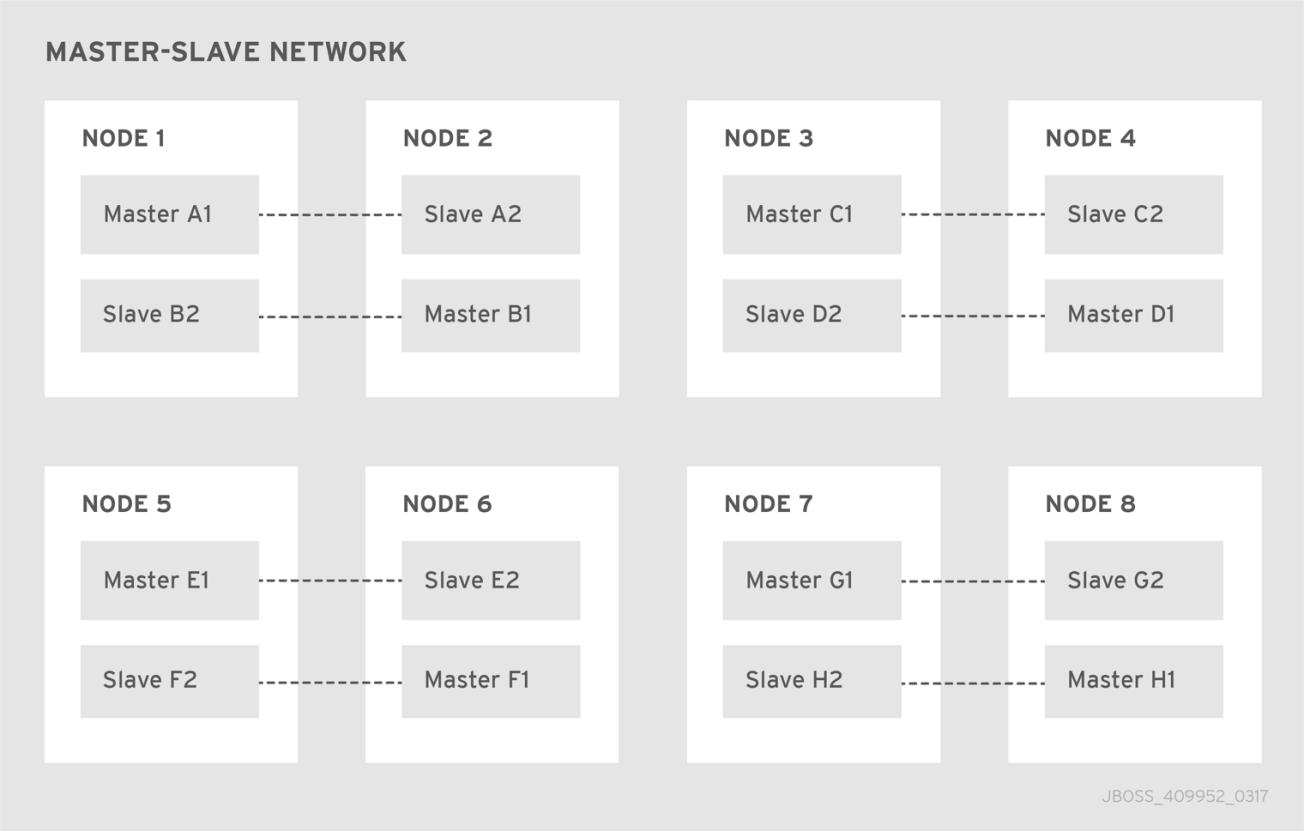
図3.3 マスター/スレーブペア



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。

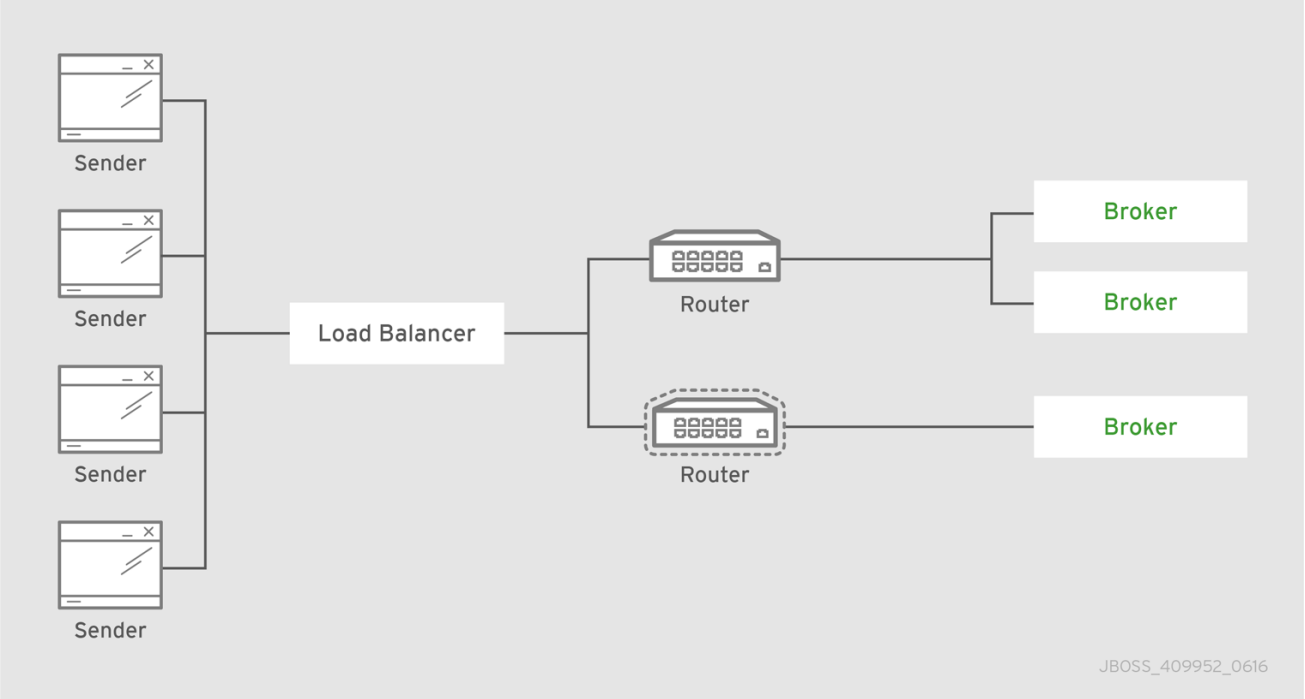
図3.4 マスター/スレーブネットワーク



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。

図3.5 ロードバランサーの背後のルーターペア

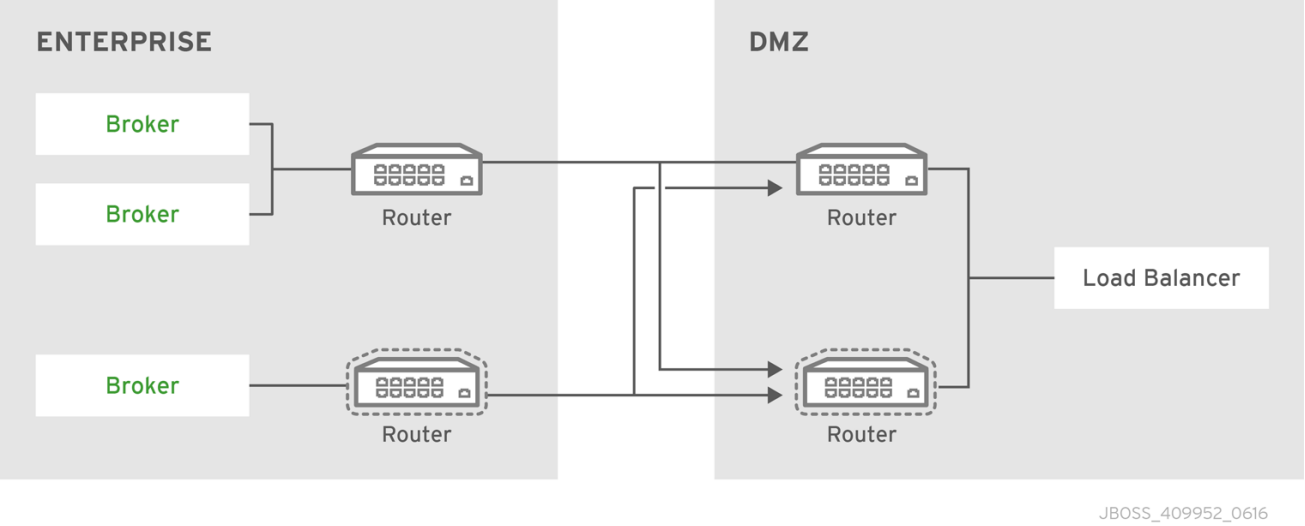


スケーラビリティーをさらに向上させるために、多くのルーター (たとえば 3 つまたは 4 つ) を使用することができます。各ルーターは、他のすべてのルーターに直接接続します。

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。

図3.6 DMZ のルーターペア



DMZ トポロジーに関する重要事項

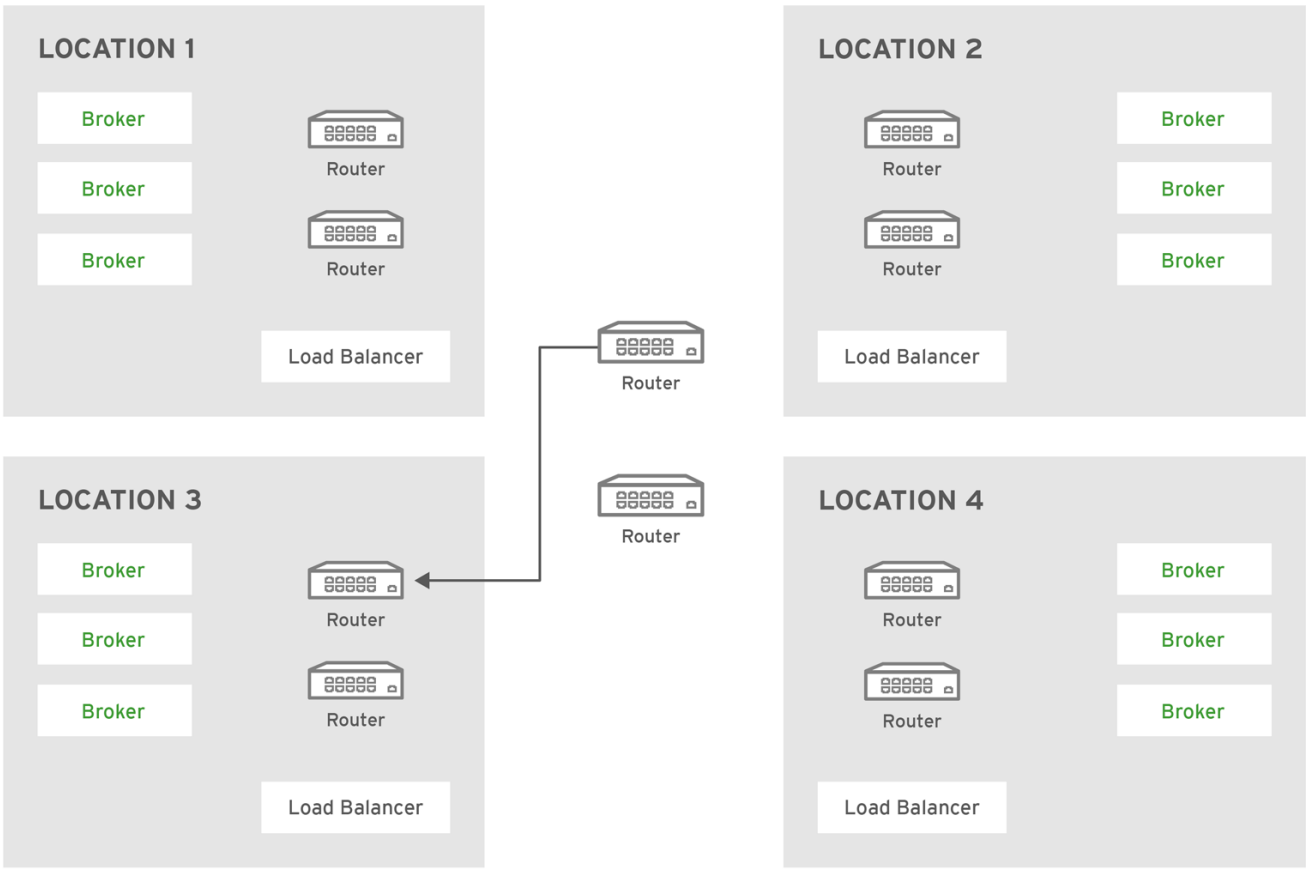
- デプロイメント内の接続のセキュリティーは、外部クライアントに使用されるセキュリティーとは異なります。たとえば、お使いのデプロイメントでは、内部のセキュリティー用にプライベート認証局 (CA) を使用して認証用に各ルーターおよびブローカーに x.509 証明書を発行する一方、外部のユーザーは別のパブリック CA を使用する可能性があります。

- 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



改訂日時： 2022-09-08 16:28:05 +1000