



**Red Hat AMQ 2021.Q3**

**AMQ Spring Boot Starter の使用**

AMQ Clients 2.10 向け



## Red Hat AMQ 2021.Q3 AMQ Spring Boot Starter の使用

---

AMQ Clients 2.10 向け

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/Using\_the\_AMQ\_Spring\_Boot\_Starter.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<sup>®</sup> is the registered trademark of Linus Torvalds in the United States and other countries.

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

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

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

Node.js<sup>®</sup> 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<sup>®</sup> 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

このガイドでは、ライブラリーをインストールして設定する方法、実践的な例を実行する方法、およびその他の AMQ コンポーネントでクライアントを使用する方法を説明します。

## 目次

多様性を受け入れるオープンソースの強化 .....	3
<b>第1章 概要</b> .....	<b>4</b>
1.1. 主な特長 .....	4
1.2. サポート対象の標準およびプロトコル .....	4
1.3. サポートされる構成 .....	4
1.4. 本書の表記慣例 .....	4
sudo コマンド .....	4
ファイルパス .....	4
変数テキスト .....	4
<b>第2章 インストール</b> .....	<b>5</b>
2.1. 前提条件 .....	5
2.2. 「USING THE RED HAT MAVEN REPOSITORY」 .....	5
2.3. 「INSTALLING A LOCAL MAVEN REPOSITORY」 .....	5
<b>第3章 スタートガイド</b> .....	<b>7</b>
3.1. 前提条件 .....	7
3.2. HELLO WORLD の実行 .....	7
サンプルの実行 .....	8
3.3. その他の例 .....	9
<b>第4章 設定</b> .....	<b>10</b>
4.1. 接続オプション .....	10
4.2. プーリングオプション .....	10
<b>付録A サブスクリプションの使用</b> .....	<b>12</b>
A.1. アカウントへのアクセス .....	12
A.2. サブスクリプションのアクティベート .....	12
A.3. リリースファイルのダウンロード .....	12
A.4. パッケージを受信するためのシステムの登録 .....	12
<b>付録B RED HAT MAVEN リポジトリの使用</b> .....	<b>14</b>
B.1. オンラインリポジトリの使用 .....	14
Maven 設定へのリポジトリの追加 .....	14
POM ファイルへのリポジトリの追加 .....	15
B.2. ローカルリポジトリの使用 .....	15
<b>付録C サンプルでの AMQ BROKER の使用</b> .....	<b>17</b>
C.1. ブローカーのインストール .....	17
C.2. ブローカーの起動 .....	17
C.3. キューの作成 .....	17
C.4. ブローカーの停止 .....	18



## 多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。これは大規模な取り組みであるため、これらの変更は今後の複数のリリースで段階的に実施されます。詳細は、[Red Hat CTO である Chris Wright のメッセージ](#)をご覧ください。

# 第1章 概要

AMQ Spring Boot Starter は、AMQ メッセージングを使用する Spring ベースのアプリケーションを作成するためのアダプターです。スタンドアロン Spring アプリケーションを構築できる Spring Boot スターターモジュールを提供します。スターターは、AMQ JMS クライアントを使用して AMQP 1.0 プロトコルを使用して通信します。

AMQ Spring Boot Starter は、複数の言語およびプラットフォームをサポートするメッセージングライブラリーのスイートである AMQ Clients の一部です。クライアントの概要は、「[AMQ Clients の概要](#)」を参照してください。本リリースに関する詳細は、『[AMQ Clients 2.10 リリースノート](#)』を参照してください。

AMQ Spring Boot Starter は [AMQP 1.0 JMS Spring Boot](#) プロジェクトをベースとしています。

## 1.1. 主な特長

- メッセージングが組み込まれたスタンドアロンの Spring アプリケーションをすばやく構築
- JMS リソースの自動設定
- JMS 接続およびセッションの設定可能なプール

## 1.2. サポート対象の標準およびプロトコル

- Spring Boot API のバージョン 2.2
- [Java Message Service](#) API のバージョン 2.0
- [Advanced Message Queueing Protocol](#) (AMQP) バージョン 1.0

## 1.3. サポートされる構成

AMQ Spring Boot Starter でサポートされている設定については、Red Hat カスタマーポータル「[Red Hat AMQ 7 でサポートされる構成](#)」を参照してください。

## 1.4. 本書の表記慣例

### sudo コマンド

本書では、root 権限を必要とするコマンドには **sudo** が使用されています。何らかの変更がシステム全体に影響する可能性があるため、**sudo** を使用する場合は注意が必要です。**sudo** の詳細は、「[sudo コマンドの使用](#)」を参照してください。

### ファイルパス

本書では、すべてのファイルパスが Linux、UNIX、および同様のオペレーティングシステムで有効です (例: `/home/andrea`)。Microsoft Windows では、同等の Windows パスを使用する必要があります (例: `C:\Users\andrea`)。

### 変数テキスト

本書には、実際の環境に固有の値に置き換える必要がある変数を含むコードブロックが含まれています。変数テキストは中括弧で囲まれ、斜体の等幅フォントとしてスタイル設定されます。たとえば、以下の例では、`<project-dir>` を実際の環境の値に置き換えます。

```
$ cd <project-dir>
```



## 第2章 インストール

本章では、環境に AMQ Spring Boot Starter をインストールする手順を説明します。

### 2.1. 前提条件

- AMQ リリースファイルおよびリポジトリにアクセスするには、[サブスクリプション](#) が必要です。
- AMQ Spring Boot Starter でプログラムをビルドするには、[Apache Maven](#) をインストールする必要があります。
- AMQ Spring Boot Starter を使用するには、Java をインストールする必要があります。

### 2.2. 「USING THE RED HAT MAVEN REPOSITORY」

Red Hat Maven リポジトリからクライアントライブラリーをダウンロードするように Maven 環境を設定します。

#### 手順

1. Red Hat リポジトリを Maven 設定または POM ファイルに追加します。設定ファイルの例については、「[オンラインリポジトリの使用](#)」を参照してください。

```
<repository>
  <id>red-hat-ga</id>
  <url>https://maven.repository.redhat.com/ga</url>
</repository>
```

2. ライブラリー依存関係を POM ファイルに追加します。

```
<dependency>
  <groupId>org.amqphub.spring</groupId>
  <artifactId>amqp-10-jms-spring-boot-starter</artifactId>
  <version>2.5.0.redhat-00001</version>
</dependency>
```

これで、クライアントが Maven プロジェクトで利用できるようになりました。

### 2.3. 「INSTALLING A LOCAL MAVEN REPOSITORY」

オンラインリポジトリの代わりに、AMQ Spring Boot Starter をファイルベースの Maven リポジトリとしてローカルファイルシステムにインストールできます。

#### 手順

1. [サブスクリプションを使用](#) して **AMQ Clients 2.10.0 Spring Boot Starter Maven リポジトリ** の .zip ファイルをダウンロードします。
2. ファイルの内容を選択したディレクトリーに展開します。  
Linux または UNIX の場合は、**unzip** コマンドを使用してファイルの内容を展開します。

```
$ unzip amq-clients-2.10.0-spring-boot-starter-maven-repository.zip
```

- Windows で、.zip ファイルを右クリックし、**Extract All** を選択します。
3. 展開したインストールディレクトリー内の **maven-repository** ディレクトリーのリポジトリーを使用するように Maven を設定します。詳細は、「[ローカルリポジトリーの使用](#)」を参照してください。

## 第3章 スタートガイド

本章では、環境を設定して簡単なメッセージングプログラムを実行する手順を説明します。

### 3.1. 前提条件

- サンプルをビルドするには、[Red Hat リポジトリ](#) または [ローカルリポジトリ](#) を使用するよう Maven を設定する必要があります。
- **localhost** で接続をリッスンするメッセージブローカーが必要です。匿名アクセスを有効にする必要があります。詳細は、「[ブローカーの開始](#)」を参照してください。
- **example** という名前のキューが必要です。詳細は、「[キューの作成](#)」を参照してください。

### 3.2. HELLO WORLD の実行

Hello World の例では、ブローカーへの接続を作成し、グリーティングが含まれるメッセージを **example** キューに送信し、それを受け取ります。成功すると、受け取ったメッセージをコンソールに出力します。

例: "Hello World!" の送受信 (**HelloWorld.java**)

```
package net.example;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.jms.annotation.EnableJms;
import org.springframework.jms.annotation.JmsListener;
import org.springframework.jms.core.JmsTemplate;

@EnableJms
@SpringBootApplication
public class HelloWorld implements CommandLineRunner {
    @Autowired
    private JmsTemplate jmsTemplate;

    public static void main(String[] args) {
        SpringApplication.run(HelloWorld.class, args);
    }

    @Override
    public void run(String... strings) throws Exception {
        sendMessage("Hello World!");
    }

    public void sendMessage(String text) {
        System.out.println(String.format("Sending '%s'", text));
        this.jmsTemplate.convertAndSend("example", text);
    }

    @JmsListener(destination = "example")
    public void receiveMessage(String text) {
```

```
    System.out.println(String.format("Received '%s'", text));  
  }  
}
```

## サンプルの実行

サンプルプログラムをコンパイルして実行するには、次の手順を使用します。

## 手順

1. 新しいプロジェクトディレクトリーを作成します。これは、以降の手順で **<project-dir>** と呼ばれます。
2. サンプル一覧を以下の場所にコピーします。

```
<project-dir>/src/main/java/net/example/HelloWorld.java
```

3. テキストエディターを使用して、新しい **<project-dir>/pom.xml** ファイルを作成します。以下の XML を追加します。

```
<project>  
  <modelVersion>4.0.0</modelVersion>  
  
  <groupId>net.example</groupId>  
  <artifactId>example</artifactId>  
  <version>1.0.0-SNAPSHOT</version>  
  
  <parent>  
    <groupId>org.springframework.boot</groupId>  
    <artifactId>spring-boot-starter-parent</artifactId>  
    <version>2.5.0</version>  
  </parent>  
  
  <dependencies>  
    <dependency>  
      <groupId>org.amqphub.spring</groupId>  
      <artifactId>amqp-10-jms-spring-boot-starter</artifactId>  
      <version>2.5.0.redhat-00001</version>  
    </dependency>  
  </dependencies>  
  
  <build>  
    <plugins>  
      <plugin>  
        <groupId>org.springframework.boot</groupId>  
        <artifactId>spring-boot-maven-plugin</artifactId>  
      </plugin>  
    </plugins>  
  </build>  
</project>
```

4. プロジェクトディレクトリーに移動し、**mvn** コマンドを使用してプログラムをコンパイルします。

```
$ mvn clean package
```

5. **java** コマンドを使用してプログラムを実行します。

```
$ java -jar target/example-1.0.0-SNAPSHOT.jar
```

Hello World の例を実行すると、以下のコンソール出力が表示されます。

```
$ java -jar target/example-1.0.0-SNAPSHOT.jar
```

```
 .  _ _ _ _ _  _  _ _ _ _  \ \ \ \
  \ \ /  _ _ ' _ _ _ _  ( )  _ _  _ _  \ \ \ \
 ( ( ) \ _ _ | ' _ | ' _ | | ' _ V _ ` | \ \ \ \
  \ \ _ _ ) | | | | | | | | | | ( | | ) ) ) )
  ' | _ _ | . _ | | | | | | | \ _ , | / / / /
  =====|_|=====|_|/=/ / / / /
```

```
:: Spring Boot ::      (v2.0.5.RELEASE)

[...]  
2018-11-05 16:21:09.849 INFO 14805 --- [main] net.example>HelloWorld: Started  
HelloWorld in 1.074 seconds (JVM running for 1.38)  
Sending 'Hello World!'  
Received 'Hello World!'
```

### 3.3. その他の例

他のプログラムの例は、[AMQP 1.0 JMS Spring Boot プロジェクト](#) および [Spring プロジェクト](#) から入手できます。

## 第4章 設定

以下のオプションは **application-properties** ファイルで使用して、Spring Boot アプリケーションを設定できます。

### 4.1. 接続オプション

これらのオプションは、AMQ Spring Boot Starter がリモート AMQP ピアへの接続を確立する方法を決定します。スターターは、AMQ JMS を使用してネットワーク上で通信します。詳細は、「[AMQ JMS クライアントの使用](#)」を参照してください。

#### **amqphub.amqp10jms.remoteUrl**

AMQ JMS クライアントが新しい接続を確立するために使用する接続 URI。

接続 URI 形式

```
amqp[s]://host:port[?option=value[&option2=value...]]
```

詳細は、『AMQ JMS クライアントの使用』の「[接続 URI](#)」を参照してください。

#### **amqphub.amqp10jms.username**

接続の認証に使用されるユーザー名。

#### **amqphub.amqp10jms.password**

接続の認証に使用されるパスワード。

#### **amqphub.amqp10jms.clientId**

接続に適用されるクライアント ID。

#### **amqphub.amqp10jms.receiveLocalOnly**

これを有効にすると、timeout 引数を指定した **receive** の呼び出しは、コンシューマーのローカルメッセージバッファーのみを確認します。それ以外の場合は、リモートピアをチェックし、利用可能なメッセージがないことを確認することができます。これはデフォルトで無効にされます。

#### **amqphub.amqp10jms.receiveNoWaitLocalOnly**

有効にすると、**receiveNoWait** に対する呼び出しは、コンシューマーのローカルメッセージバッファーのみを確認します。それ以外の場合は、リモートピアをチェックし、利用可能なメッセージがないことを確認することができます。これはデフォルトで無効にされます。

### 4.2. プーリングオプション

これらのオプションは、AMQ Spring Boot Starter が JMS 接続およびセッションをキャッシュする方法を決定します。スターターは、プーリングに AMQ JMS プールを使用します。詳細は、「[AMQ JMS Pool Library の使用](#)」を参照してください。

#### **amqphub.amqp10jms.pool.enabled**

プーリングを有効にするかどうかを制御します。これはデフォルトで無効にされます。

#### **amqphub.amqp10jms.pool.maxConnections**

1つのプールの最大接続数。デフォルトは1です。

#### **amqphub.amqp10jms.pool.maxSessionsPerConnection**

各接続のセッションの最大数。デフォルトは500です。負の値を指定すると制限が削除されます。

制限を超えた場合、**createSession()** は設定に応じて例外をブロックするか、または例外を出力しません。

#### **amqphub.amqp10jms.pool.blockIfSessionPoolsFull**

有効な場合は、プールでセッションが利用可能になるまで **createSession()** ブロックを呼び出します。これはデフォルトで有効になっています。

無効な場合は、**createSession()** に対する呼び出しは、セッションが利用できない場合に **IllegalStateException** を出力します。

#### **amqphub.amqp10jms.pool.blockIfSessionPoolsFullTimeout**

**createSession()** へのブロックされた呼び出しが **IllegalStateException** を出力するまでの時間 (ミリ秒単位)。デフォルトは -1 で、呼び出しが永久にブロックされることを意味します。

#### **amqphub.amqp10jms.pool.connectionIdleTimeout**

現在貸し出されていない接続をプールから削除できるようになるまでのミリ秒単位の時間。デフォルトは 30 秒です。値 0 を指定すると、タイムアウトが無効になります。

#### **amqphub.amqp10jms.pool.connectionCheckInterval**

期限切れ接続の定期的なチェックの間隔 (ミリ秒単位)。デフォルトは 0 で、チェックが無効になっていることを意味します。

#### **amqphub.amqp10jms.pool.useAnonymousProducers**

これを有効にすると、**createProducer()** へのすべての呼び出しに単一の匿名 JMS **MessageProducer** を使用します。これはデフォルトで有効になっています。

まれに、この動作が望ましくない場合があります。これを無効にすると、**createProducer()** へのすべての呼び出しは新しい **MessageProducer** インスタンスになります。

#### **amqphub.amqp10jms.pool.explicitProducerCacheSize**

匿名プロデューサーを使用しない場合、JMS **Session** は明示的な宛先で、一定数の **MessageProducer** オブジェクトをキャッシュするように設定できます。キャッシュされたプロデューサーに一致しない新しいプロデューサーが作成されると、キャッシュ内の最も古いエンターリーがエビクトされます。

#### **amqphub.amqp10jms.pool.useProviderJMSContext**

これを有効にすると、基礎となる JMS プロバイダーの **JMSContext** クラスを使用します。これはデフォルトで無効にされます。

通常、プールは独自の汎用 **JMSContext** 実装を使用して、プロバイダー実装ではなく、プールから接続をラップします。汎用の実装には、プロバイダーの実装が制限されない可能性があります。ただし、有効にすると、**JMSContext** API からの接続はプールによって管理されません。

## 付録A サブスクリプションの使用

AMQ は、ソフトウェアサブスクリプションから提供されます。サブスクリプションを管理するには、Red Hat カスタマーポータルでアカウントにアクセスします。

### A.1. アカウントへのアクセス

#### 手順

1. [access.redhat.com](https://access.redhat.com) に移動します。
2. アカウントがない場合は、作成します。
3. アカウントにログインします。

### A.2. サブスクリプションのアクティベート

#### 手順

1. [access.redhat.com](https://access.redhat.com) に移動します。
2. サブスクリプション に移動します。
3. **Activate a subscription** に移動し、16桁のアクティベーション番号を入力します。

### A.3. リリースファイルのダウンロード

.zip、.tar.gz、およびその他のリリースファイルにアクセスするには、カスタマーポータルを使用してダウンロードする関連ファイルを検索します。RPM パッケージまたは Red Hat Maven リポジトリを使用している場合、この手順は必要ありません。

#### 手順

1. ブラウザーを開き、[access.redhat.com/downloads](https://access.redhat.com/downloads) で Red Hat カスタマーポータルの **Product Downloads** ページにログインします。
2. **INTEGRATION AND AUTOMATION** カテゴリで **Red Hat AMQ** エントリーを見つけます。
3. 必要な AMQ 製品を選択します。 **Software Downloads** ページが開きます。
4. コンポーネントの **Download** リンクをクリックします。

### A.4. パッケージを受信するためのシステムの登録

この製品の RPM パッケージを Red Hat Enterprise Linux にインストールするには、お使いのシステムを登録する必要があります。ダウンロードしたリリースファイルを使用している場合は、この手順は必要ありません。

#### 手順

1. [access.redhat.com](https://access.redhat.com) に移動します。
2. **Registration Assistant** に移動します。



3. ご使用の OS バージョンを選択し、次のページに進みます。
4. システムの端末に一覧表示されたコマンドを使用して、登録を完了します。

システムを登録する方法は、以下のリソースを参照してください。

- [Red Hat Enterprise Linux 7 - システム登録およびサブスクリプション管理](#)
- [Red Hat Enterprise Linux 8 - システム登録およびサブスクリプション管理](#)

## 付録B RED HAT MAVEN リポジトリの使用

本セクションでは、ソフトウェアで Red Hat が提供する Maven リポジトリを使用する方法を説明します。

### B.1. オンラインリポジトリの使用

Red Hat は、Maven ベースのプロジェクトで使用する中央 Maven リポジトリを維持します。詳細は、[リポジトリの welcome ページ](#) を参照してください。

Red Hat リポジトリを使用するように Maven を設定する方法は 2 つあります。

- [Maven 設定にリポジトリを追加する](#)
- [リポジトリを POM ファイルに追加する](#)

#### Maven 設定へのリポジトリの追加

この設定の手法は、POM ファイルがリポジトリ設定を上書きせず、含まれるプロファイルが有効になっている限り、ユーザーが所有するすべての Maven プロジェクトに適用されます。

#### 手順

1. Maven **settings.xml** ファイルを見つけます。通常、これはユーザーのホームディレクトリ内の **.m2** ディレクトリ内にあります。ファイルが存在しない場合は、テキストエディターを使用して作成します。

Linux または UNIX の場合:

```
/home/<username>/.m2/settings.xml
```

Windows の場合:

```
C:\Users\<username>\.m2\settings.xml
```

2. 以下の例のように、Red Hat リポジトリを含む新しいプロファイルを **settings.xml** ファイルの **profiles** 要素に追加します。

例: Red Hat リポジトリが含まれる Maven **settings.xml** ファイル

```
<settings>
  <profiles>
    <profile>
      <id>red-hat</id>
      <repositories>
        <repository>
          <id>red-hat-ga</id>
          <url>https://maven.repository.redhat.com/ga</url>
        </repository>
      </repositories>
      <pluginRepositories>
        <pluginRepository>
          <id>red-hat-ga</id>
          <url>https://maven.repository.redhat.com/ga</url>
          <releases>
            <enabled>true</enabled>
          </releases>
        </pluginRepository>
      </pluginRepositories>
    </profile>
  </profiles>
</settings>
```

```

    </releases>
    <snapshots>
      <enabled>>false</enabled>
    </snapshots>
  </pluginRepository>
</pluginRepositories>
</profile>
</profiles>
<activeProfiles>
  <activeProfile>red-hat</activeProfile>
</activeProfiles>
</settings>

```

Maven 設定に関する詳細は、[Maven 設定リファレンス](#) を参照してください。

### POM ファイルへのリポジトリの追加

プロジェクトに直接リポジトリを設定するには、以下の例のように、POM ファイルの **repositories** 要素に新しいエントリを追加します。

例: Red Hat リポジトリが含まれる Maven pom.xml ファイル

```

<project>
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.example</groupId>
  <artifactId>example-app</artifactId>
  <version>1.0.0</version>

  <repositories>
    <repository>
      <id>red-hat-ga</id>
      <url>https://maven.repository.redhat.com/ga</url>
    </repository>
  </repositories>
</project>

```

POM ファイル設定の詳細は、「[Maven POM リファレンス](#)」を参照してください。

## B.2. ローカルリポジトリの使用

Red Hat は、そのコンポーネントの一部に対してファイルベースの Maven リポジトリを提供します。これらは、ローカルファイルシステムに抽出できるダウンロード可能なアーカイブとして提供されます。

ローカルに抽出したリポジトリを使用するように Maven を設定するには、Maven 設定または POM ファイルに以下の XML を適用します。

```

<repository>
  <id>red-hat-local</id>
  <url>${repository-url}</url>
</repository>

```

**\${repository-url}** 展開したリポジトリのローカルファイルシステムパスを含むファイルの URL でなければなりません。

表B.1 ローカル Maven リポジトリーの URL の例

オペレーティングシステム	ファイルシステムパス	URL
Linux または UNIX	<b>/home/alice/maven-repository</b>	<b>file:/home/alice/maven-repository</b>
Windows	<b>C:\repos\red-hat</b>	<b>file:C:\repos\red-hat</b>

## 付録C サンプルでの AMQ BROKER の使用

AMQ Spring Boot Starter の例では、**example** という名前のキューが含まれる実行中のメッセージブローカーが必要です。以下の手順に従って、ブローカーをインストールして起動し、キューを定義します。

### C.1. ブローカーのインストール

『AMQ Broker の使用』の説明に従い [ブロッカーをインストール](#) して、[ブローカーインスタンスを作成](#) します。匿名アクセスを有効にします。

以下の手順では、**<broker-instance-dir>** としてブローカーインスタンスの場所を参照します。

### C.2. ブローカーの起動

#### 手順

1. **artemis run** コマンドを使用してブローカーを起動します。

```
$ <broker-instance-dir>/bin/artemis run
```

2. コンソールの出力で、起動時にログに記録される重要なエラーの有無を確認します。ブローカーは、準備が整う際に **Server is now live** をログに記録します。

```
$ example-broker/bin/artemis run
```

```
  ____
 /    \ / \  / \ / \  / \  / \  / \  / \  / \  / \  / \  / \  / \
/      /   /  /   /   /   /   /   /   /   /   /   /   /   /
/      /   /  /   /   /   /   /   /   /   /   /   /   /   /
/      /   /  /   /   /   /   /   /   /   /   /   /   /   /
/      /   /  /   /   /   /   /   /   /   /   /   /   /   /
/      /   /  /   /   /   /   /   /   /   /   /   /   /   /
```

```
Red Hat AMQ <version>
```

```
2020-06-03 12:12:11,807 INFO [org.apache.activemq.artemis.integration.bootstrap]
AMQ101000: Starting ActiveMQ Artemis Server
```

```
...
```

```
2020-06-03 12:12:12,336 INFO [org.apache.activemq.artemis.core.server] AMQ221007:
Server is now live
```

```
...
```

### C.3. キューの作成

新しいターミナルで、**artemis queue** コマンドを使用して **example** という名前のキューを作成します。

```
$ <broker-instance-dir>/bin/artemis queue create --name example --address example --auto-
create-address --anycast
```

yes または no の質問への回答を求めるプロンプトが表示されます。そのすべてに no (**N**) と回答します。

キューが作成されると、ブローカーはサンプルプログラムと使用できるようになります。

## C.4. ブローカーの停止

サンプルの実行が終了したら、**artemis stop** コマンドを使用してブローカーを停止します。

```
$ <broker-instance-dir>/bin/artemis stop
```

改訂日時: 2021-08-29 15:42:47 +1000