Red Hat AMQ Broker 7.11

AMQ Broker 시작하기

AMQ Broker 7.11과 함께 사용하는 경우

Last Updated: 2024-06-11

Red Hat AMQ Broker 7.11 AMQ Broker 시작하기

AMQ Broker 7.11과 함께 사용하는 경우

법적 공지

초록

이 가이드에서는 AMQ Broker 사용을 시작하는 방법을 설명합니다.

차례

보다 포괄적 수용을 위한 오픈 소스 용어 교체

Red Hat은 코드, 문서, 웹 속성에서 문제가 있는 용어를 교체하기 위해 최선을 다하고 있습니다. 먼저 마스터(master), 슬레이브(slave), 블랙리스트(blacklist), 화이트리스트(whitelist) 등 네 가지 용어를 교체하고 있습니다. 이러한 변경 작업은 작업 범위가 크므로 향후 여러 릴리스에 걸쳐 점차 구현할 예정입니다. 자세한 내용은 CTO Chris Wright의 메시지 를 참조하십시오.

1장. 개요

AMQ Broker는 ActiveMQ Artemis를 기반으로 하는 고성능 메시징 구현입니다. 메시지 지속성을 위해 비동기 저널을 사용하며 여러 언어, 프로토콜 및 플랫폼을 지원합니다.

1.1. 주요 기능

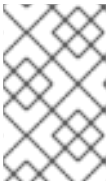
AMQ Broker는 다음 기능을 제공합니다.

- 클러스터링 및 고가용성 옵션
- 빠르고 네이티브 IO 지속성
- 로컬 트랜잭션 지원
- AMQ Core Protocol JMS 및 AMQ OpenECDHEre JMS 클라이언트를 사용할 때 XA 트랜잭션 지원
- 광범위한 플랫폼 지원을 위해 Java로 작성
- 다중 관리 인터페이스: AMQ Management Console, 관리 API 및 ScanSetting

1.2. 지원되는 표준 및 프로토콜

AMQ Broker는 다음 표준 및 프로토콜을 지원합니다.

- Wire 프로토콜:
  - 코어 프로토콜
  - AMQP 1.0
  - MQTT
  - OpenECDHEre (A-MQ 6 클라이언트에서 사용)
  - STOMP
- JMS 2.0



참고

AMQP 내의 분산 트랜잭션(XA) 세부 정보는 사양의 1.0 버전에서 제공되지 않습니다. 환경에서 분산 트랜잭션을 지원해야 하는 경우 AMQ Core Protocol JMS를 사용하는 것이 좋습니다.

1.3. 지원되는 구성

AMQ Broker 지원 구성에 대한 최신 정보는Red Hat AMQ 7 지원 구성 "Red Hat Customer Portal의 지원 구성 문서를 참조하십시오.

1.4. 문서 규칙

이 문서에서는 **sudo** 명령, 파일 경로 및 대체 가능한 값에 대해 다음 규칙을 사용합니다.

sudo 명령

이 문서에서는 root 권한이 필요한 모든 명령에 **sudo** 를 사용합니다. **sudo** 를 사용할 때는 변경 사항이 전체 시스템에 영향을 줄 수 있으므로 항상 주의해야 합니다.

sudo 사용에 대한 자세한 내용은 sudo 액세스 관리를 참조하십시오.

이 문서에서 파일 경로 사용 정보

이 문서에서 모든 파일 경로는 Linux, UNIX 및 유사한 운영 체제(예: **/home/**..)에 유효합니다. Microsoft Windows를 사용하는 경우 동등한 Microsoft Windows 경로(예 **: C:\Users**..)를 사용해야 합니다.

교체 가능한 값

이 문서에서는 환경에 고유한 값으로 교체해야 하는 대체 가능한 값을 사용하는 경우가 있습니다. 교체할 수 있는 값은 소문자로 묶고(< >), **italics 및 monospace 글꼴을 사용하여 스타일을 지정합니다. 여러 단어가 밑줄(_)으로 구분됩니다.**

예를 들어 다음 명령에서 < *install_dir*>*을 사용자* 고유의 디렉터리 이름으로 바꿉니다.

```
$ <install_dir>/bin/artemis create mybroker
```

2장. AMQ BROKER 이해

AMQ Broker를 사용하면 이기종 시스템을 함께 느슨하게 결합하는 동시에 신뢰성, 트랜잭션 및 기타 여러 기능을 제공할 수 있습니다. **AMQ Broker**를 사용하기 전에 먼저 제공하는 기능을 이해해야 합니다.

2.1. 브로커 인스턴스

AMQ Broker에서 설치된 **AMQ Broker** 소프트웨어는 하나 이상의 *브로커 인스턴스의 경우* "홈"으로 사용됩니다. 이 아키텍처는 다음과 같은 몇 가지 이점을 제공합니다.

- 단일 **AMQ Broker** 설치에서 필요한 만큼의 브로커 인스턴스를 생성할 수 있습니다.

  AMQ Broker 설치에는 각 브로커 인스턴스를 실행하는 데 필요한 바이너리 및 리소스가 포함되어 있습니다. 그런 다음 이러한 리소스는 브로커 인스턴스 간에 공유됩니다.

- **AMQ Broker**의 새 버전으로 업그레이드할 때 해당 호스트에서 여러 브로커 인스턴스를 실행하는 경우에도 소프트웨어를 한 번만 업데이트하면 됩니다.

브로커 인스턴스를 메시지 브로커로 생각할 수 있습니다. 각 브로커 인스턴스에는 고유한 구성 및 런타임 데이터가 포함된 자체 디렉터리가 있습니다. 이 런타임 데이터는 로그 및 데이터 파일로 구성되며 호스트의 고유 브로커 프로세스와 연결됩니다.

2.2. 메시지 지속성

AMQ Broker는 브로커가 실패하거나 예기치 않게 종료해도 메시지 데이터를 유지하여 메시지가 손실되지 않도록 합니다. **AMQ Broker**는 메시지 지속성을 위해 저널 기반 지속성 및 데이터베이스 지속성이라는 두 가지 옵션을 제공합니다.

journal-based persistence

기본 방법인 이 옵션은 메시지 데이터를 파일 시스템에 저장된 메시지 저널 파일에 씁니다. 처음에는 각 저널 파일이 고정된 크기로 자동으로 생성되고 빈 데이터로 채워집니다. 클라이언트가 다양한 브로커 작업을 수행할 때 레코드가 저널에 추가됩니다. 저널 파일 중 하나가 가득 차면 브로커가 다음 저널 파일로 이동합니다.

journalctl 기반 지속성은 로컬 및 **XA** 트랜잭션을 모두 포함하여 트랜잭션 작업을 지원합니다.

ScanSetting 기반 지속성을 사용하려면 파일 시스템에 IO 인터페이스가 필요합니다. AMQ Broker는 다음을 지원합니다.

Linux 비동기 IO(AIO)

AIO는 일반적으로 최상의 성능을 제공합니다. Red Hat AMQ 7 지원 구성에서 네트워크 파일 시스템이 지원되는 것으로 나열되어 있는지 확인합니다.

Java NIO

Java NIO는 우수한 성능을 제공하며 Java 6 이상 런타임을 사용하는 모든 플랫폼에서 실행할 수 있습니다.

데이터베이스 지속성

이 옵션은 JDBC(Java Database Connectivity)를 사용하여 메시지 및 바인딩 데이터를 데이터베이스에 저장합니다. 현재 환경에 안정적이고 고성능 데이터베이스 플랫폼을 보유하고 있거나 데이터베이스 사용이 회사 정책에 따라 필요한 경우 이 옵션을 사용하는 것이 좋습니다.

브로커 JDBC 지속성 저장소는 표준 JDBC 드라이버를 사용하여 데이터베이스 테이블에 메시지 및 바인딩 데이터를 저장하는 JDBC 연결을 생성합니다. 데이터베이스 테이블의 데이터는 저널 기반 지속성과 동일한 인코딩을 사용하여 인코딩됩니다. 즉, 데이터베이스에 저장된 메시지는 SQL을 사용하여 직접 액세스하는 경우 사람이 읽을 수 없습니다.

데이터베이스 지속성을 사용하려면 지원되는 데이터베이스 플랫폼을 사용해야 합니다. 현재 지원되는 데이터베이스 플랫폼을 보려면 Red Hat AMQ 7 지원 구성을 참조하십시오.

2.3. 리소스 사용

AMQ Broker는 브로커의 메모리 및 리소스 사용을 제한하는 다양한 옵션을 제공합니다.

리소스 제한

각 사용자에 대한 연결 및 대기열 제한을 설정할 수 있습니다. 이로 인해 사용자가 브로커의 리소스를 너무 많이 소비하고 다른 사용자에게 성능이 저하되지 않습니다.

메시지 페이징

메시지 페이징을 사용하면 AMQ Broker가 메모리가 제한된 상태에서 수백만 개의 메시지를 포함하는 대규모 큐를 지원할 수 있습니다. 브로커가 메모리 용량을 초과하는 메시지 급증을 수신하면 디스크에 페이징 메시지를 시작합니다. 이 페이징 프로세스는 투명합니다. 브로커는 필요에 따라 메모리 내외로 메시지를 표시합니다.

메시지 페이징은 주소 기반입니다. 주소에 대한 메모리에 있는 모든 메시지의 크기가 최대 크기를 초과하면 주소에 대한 각 추가 메시지가 주소의 페이지 파일로 페이징됩니다.

큰 메시지

AMQ Broker를 사용하면 제한된 메모리 리소스로 실행되는 경우에도 대규모 메시지를 보내고 받을 수 있습니다. 메모리에 대용량 메시지를 저장하는 오버헤드를 방지하기 위해 이러한 큰 메시지를 파일 시스템 또는 데이터베이스 테이블에 저장하도록 AMQ 브로커를 구성할 수 있습니다.

2.4. 모니터링 및 관리

AMQ Broker는 브로커를 모니터링하고 관리하는 데 사용할 수 있는 여러 도구를 제공합니다.

AMQ 관리 콘솔

AMQ Management Console은 웹 브라우저를 통해 액세스할 수 있는 웹 인터페이스입니다. 을 사용하여 네트워크 상태를 모니터링하고, 브로커 토폴로지를 보고, 브로커 리소스를 생성 및 삭제할 수 있습니다.

CLI

AMQ Broker는 브로커를 관리하는 데 사용할 수 있는 artemis CLI를 제공합니다. CLI를 사용하면 브로커 인스턴스를 생성, 시작 및 중지할 수 있습니다. CLI는 메시지 저널을 관리하기 위한 여러 명령도 제공합니다.

관리 API

AMQ Broker는 광범위한 관리 API를 제공합니다. 이를 사용하여 브로커의 구성을 수정하고, 새 리소스를 생성하고, 이러한 리소스를 검사하고, 상호 작용할 수 있습니다. 클라이언트는 관리 API를 사용하여 브로커를 관리하고 관리 알림을 구독할 수도 있습니다.

AMQ Broker는 관리 API를 사용하기 위해 다음과 같은 방법을 제공합니다.

- Java Management Extensions(JMX) - Java 애플리케이션 관리를 위한 표준 기술입니다. 브로커의 관리 작업은 AMQECDHEs 인터페이스를 통해 노출됩니다.
- JMS API - 관리 작업은 표준 JMS 메시지를 사용하여 특수 관리 JMS 큐로 전송됩니다.

로그

각 브로커 인스턴스는 오류 메시지, 경고 및 기타 브로커 관련 정보 및 활동을 기록합니다. 로깅 수준, 로그 파일의 위치 및 로그 형식을 구성할 수 있습니다. 그런 다음 결과 로그 파일을 사용하여 브로

커를 모니터링하고 오류 조건을 진단할 수 있습니다.

3장. AMQ BROKER 설치

AMQ Broker는 플랫폼 독립적인 아카이브 파일로 배포됩니다. 시스템에 AMQ Broker를 설치하려면 아카이브를 다운로드하고 콘텐츠를 추출해야 합니다. 또한 아카이브에 포함된 디렉터리를 이해해야 합니다.

사전 요구 사항

- AMQ Broker를 설치하는 호스트는 AMQ Broker 지원 구성을 충족해야 합니다.

  자세한 내용은 Red Hat AMQ 7 지원 구성을 참조하십시오.

3.1. AMQ BROKER 아카이브 다운로드

AMQ Broker는 플랫폼 독립적인 아카이브 파일로 배포됩니다. Red Hat 고객 포털에서 다운로드할 수 있습니다.

사전 요구 사항

- Red Hat 서브스크립션이 있어야 합니다.

  자세한 내용은 서브스크립션 사용을 참조하십시오.

절차

1. 웹 브라우저에서 https://access.redhat.com/downloads/ 으로 이동하여 로그인합니다.

   제품 다운로드 페이지가 표시됩니다.

2. JBoss Integration and Automation 섹션에서 Red Hat AMQ Broker 링크를 클릭합니다.

   Software Downloads 페이지가 표시됩니다.

3. 버전 드롭다운 메뉴에서 원하는 AMQ Broker 버전을 선택합니다.

4. 릴리스 탭에서 다운로드 하려는 특정 **AMQ Broker** 파일에 대한 다운로드 링크를 클릭합니다.

3.2. LINUX에서 AMQ BROKER 아카이브 추출

Red Hat Enterprise Linux에 **AMQ Broker**를 설치하는 경우 **AMQ Broker**를 위한 새 사용자 계정을 생성한 다음 설치 아카이브에서 콘텐츠를 추출합니다.

절차

1. **amq-broker** 라는 새 사용자를 생성하고 암호를 제공합니다.

```
$ sudo useradd amq-broker
$ sudo passwd amq-broker
```

2. **/opt/redhat/amq-broker** 디렉터리를 만들고 새 **amq-broker** 사용자 및 그룹을 소유자로 설정합니다.

```
$ sudo mkdir /opt/redhat
$ sudo mkdir /opt/redhat/amq-broker
$ sudo chown -R amq-broker:amq-broker /opt/redhat/amq-broker
```

3. 아카이브의 소유자를 새 사용자로 변경합니다.

```
$ sudo chown amq-broker:amq-broker amq-broker-7.x.x-bin.zip
```

4. 방금 만든 디렉터리로 설치 아카이브를 이동합니다.

```
$ sudo mv amq-broker-7.x.x-bin.zip /opt/redhat/amq-broker
```

5. 새 **amq-broker** 사용자로 **unzip** 명령을 사용하여 콘텐츠를 추출합니다.

```
$ su - amq-broker
$ cd /opt/redhat/amq-broker
$ unzip <archive_name>.zip
$ exit
```

amq-broker-7.11 과 유사한 이름을 가진 디렉토리가 생성됩니다. 문서에서 이 위치는 *<install_dir>*이라고 합니다.

3.3. WINDOWS 시스템에서 AMQ BROKER 아카이브 추출

Windows 시스템에 AMQ Broker를 설치하는 경우 AMQ Broker에 대한 새 디렉터리 폴더를 생성한 다음 컨텐츠를 추출합니다.

절차

1. Windows Explorer를 사용하여 원하는 드라이브 문자에 디렉터리 폴더 \redhat\amq-broker 를 생성합니다.

   예: C:\redhat\amq-broker

2. Windows Explorer를 사용하여 방금 만든 디렉터리로 설치 아카이브를 이동합니다.

3. \redhat\amq-broker 디렉터리에서 설치 아카이브 zip 파일을 마우스 오른쪽 버튼으로 클릭하고 Extract All.

   amq-broker-7.11 과 유사한 이름을 가진 디렉토리가 생성됩니다. 문서에서 이 위치는 < *install_dir>이라고 합니다*.

3.4. AMQ BROKER 설치 아카이브 콘텐츠 이해

아카이브를 추출하여 생성된 디렉터리는 AMQ Broker 설치의 최상위 디렉터리입니다. 이 디렉터리는 < *install_dir>*이라고 하며 다음 내용이 포함됩니다.

이 디렉터리...	contains...
***<install_dir>*/web/api**	API 설명서.
***<install_dir>*/bin**	AMQ Broker를 실행하는 데 필요한 바이너리 및 스크립트입니다.
***<install_dir>*/etc**	구성 파일.
***<install_dir>*/examples**	JMS 및 Java EE 예제.

이 디렉터리...	contains...
***<install_dir>*/lib**	AMQ Broker를 실행하는 데 필요한 ScanSetting 및 라이브러리.
***<install_dir>*/schema**	AMQ Broker 구성의 유효성을 확인하는 데 사용되는 XML 스키마입니다.
***<install_dir>*/web**	AMQ Broker를 실행할 때 로드된 웹 컨텍스트입니다.

4장. 독립 실행형 브로커 생성

AMQ Broker를 사용하여 로컬 시스템에 독립 실행형 브로커 인스턴스를 생성하고 시작하며 일부 테스트 메시지를 생성 및 사용할 수 있습니다.

사전 요구 사항

- **AMQ Broker**가 설치되어 있어야 합니다.

  자세한 내용은 *3장. AMQ Broker 설치*의 내용을 참조하십시오.

4.1. 브로커 인스턴스 생성

브로커 인스턴스는 브로커의 구성 및 런타임 데이터가 포함된 디렉터리입니다. 새 브로커 인스턴스를 생성하려면 먼저 **broker** 인스턴스의 디렉터리를 생성한 다음 **artemis create** 명령을 사용하여 브로커 인스턴스를 생성합니다.

다음 절차에서는 로컬 머신에 간단한 독립 실행형 브로커를 생성하는 방법을 설명합니다. 브로커는 기본 기본 기본 구성을 사용하고 지원되는 메시징 프로토콜을 사용하여 클라이언트의 연결을 허용합니다.

절차

1. **broker** 인스턴스에 사용할 디렉터리를 생성합니다.

If you are using...	이 작업을 수행합니다.

Red Hat Enterprise Linux	1. broker 인스턴스의 위치로 사용할 새 디렉터리를 생성합니다.
	`$ sudo mkdir /var/opt/amq-broker`
	2. 설치 중에 생성한 사용자를 할당합니다.
	`$ sudo chown -R amq-broker:amq-broker /var/opt/amq-broker`

Windows	Windows Explorer를 사용하여 브로커 인스턴스의 위치로 사용할 새 폴더를 만듭니다.

2.

artemis create 명령을 사용하여 브로커를 생성합니다.

If you are using...	이 작업을 수행합니다.

Red Hat Enterprise Linux	1. 설치 중에 생성한 사용자 계정으로 전환합니다.
	`$ su - amq-broker`
	2. 브로커 인스턴스에 대해 방금 생성한 디렉터리로 변경합니다.
	`$ cd /var/opt/amq-broker`
	3. broker 인스턴스의 디렉터리에서 broker 인스턴스를 생성합니다.
	`$ <install_dir>/bin/artemis create mybroker`

Windows	1. broker 인스턴스에 대해 방금 생성한 디렉터리에서 명령 프롬프트를 엽니다. 2. broker 인스턴스의 디렉터리에서 broker 인스턴스를 생성합니다. `> <install_dir>\bin\artemis.cmd create mybroker`

3.

artemis create 프롬프트에 따라 브로커 인스턴스를 구성합니다.

예 4.1. artemis create를 사용하여 브로커 인스턴스 구성

```
$ /opt/redhat/amq-broker/bin/artemis create mybroker

Creating ActiveMQ Artemis instance at: /var/opt/amq-broker/mybroker

--user: is mandatory with this configuration:
Please provide the default username:
admin

--password: is mandatory with this configuration:
Please provide the default password:

--role: is mandatory with this configuration:
Please provide the default role:
amq

--allow-anonymous | --require-login: is mandatory with this configuration:
Allow anonymous access? (Y/N):
Y

Auto tuning journal ...
done! Your system can make 19.23 writes per millisecond, your journal-buffer-
```

```
timeout will be 52000

You can now start the broker by executing:

"/var/opt/amq-broker/mybroker/bin/artemis" run

Or you can run the broker in the background using:

"/var/opt/amq-broker/mybroker/bin/artemis-service" start
```

4.2. 브로커 인스턴스 시작

브로커 인스턴스가 생성되면 **artemis run** 명령을 사용하여 시작합니다.

절차

1. 설치 중에 생성한 사용자 계정으로 전환합니다.

```
$ su - amq-broker
```

2. **artemis run** 명령을 사용하여 브로커 인스턴스를 시작합니다.

```
$ /var/opt/amq-broker/mybroker/bin/artemis run

        __  __  ____    ____          _
      /\   |  \/  |/ __ \  |  _ \        | |
     /  \  | \  / | |  | | | |_) |_ __ ___ | | _____ _ __
    / /\ \ | |\/| | |  | | |  _ <| '__/ _ \| |/ / _ \ '__|
   / ____ \| |  | | |__| | | |_) | | | (_) |   <  __/ |
  /_/    \_\_|  |_|\____/  |____/|_|  \___/|_|\_\___|_|

Red Hat JBoss AMQ 7.2.1.GA

10:53:43,959 INFO  [org.apache.activemq.artemis.integration.bootstrap] AMQ101000:
Starting ActiveMQ Artemis Server
10:53:44,076 INFO  [org.apache.activemq.artemis.core.server] AMQ221000: live Message
Broker is starting with configuration Broker Configuration
(clustered=false,journalDirectory=./data/journal,bindingsDirectory=./data/bindings,largeMessage
sDirectory=./data/large-messages,pagingDirectory=./data/paging)
10:53:44,099 INFO  [org.apache.activemq.artemis.core.server] AMQ221012: Using AIO
Journal
...
```

브로커는 다음 정보를 사용하여 로그 출력을 시작하고 표시합니다.

- 트랜잭션 로그 및 클러스터 구성의 위치입니다.
- 메시지 지속성(이 경우 AIO)에 사용되는 저널 유형입니다.
- 클라이언트 연결을 허용할 수 있는 URI입니다.

  기본적으로 포트 61616은 지원되는 모든 프로토콜(CORE, MQTT, AMQP, STOMP, HORNETQ 및 OPENWIRE)의 연결을 허용할 수 있습니다. 또한 각 프로토콜마다 별도의 개별 포트도 있습니다.
- 웹 콘솔은 http://localhost:8161 에서 사용할 수 있습니다.
- Jolokia 서비스(RR을 통한 JMX)는 http://localhost:8161/jolokia 에서 사용할 수 있습니다.

4.3. 테스트 메시지 생성 및 사용

브로커를 시작한 후 올바르게 실행되고 있는지 확인해야 합니다. 여기에는 몇 가지 테스트 메시지를 생성하고 브로커로 보낸 후 소비해야 합니다.

절차

1. artemis producer 명령을 사용하여 몇 가지 테스트 메시지를 생성하고 브로커로 보냅니다.

   이 명령은 브로커에서 자동으로 생성된 helloworld 주소로 100개의 메시지를 보냅니다. 생산자는 지원되는 모든 메시징 프로토콜을 허용하는 기본 포트 61616을 사용하여 브로커에 연결합니다.

```
$ /opt/redhat/amq-broker/amq-broker-7.2.0/bin/artemis producer --destination
helloworld --message-count 100 --url tcp://localhost:61616
Producer ActiveMQQueue[helloworld], thread=0 Started to calculate elapsed time ...

Producer ActiveMQQueue[helloworld], thread=0 Produced: 100 messages
Producer ActiveMQQueue[helloworld], thread=0 Elapsed time in second : 1 s
Producer ActiveMQQueue[helloworld], thread=0 Elapsed time in milli second : 1289
milli seconds
```

2. 웹 콘솔을 사용하여 브로커에 저장된 메시지를 확인합니다.

a. 웹 브라우저에서 http://localhost:8161 로 이동합니다.

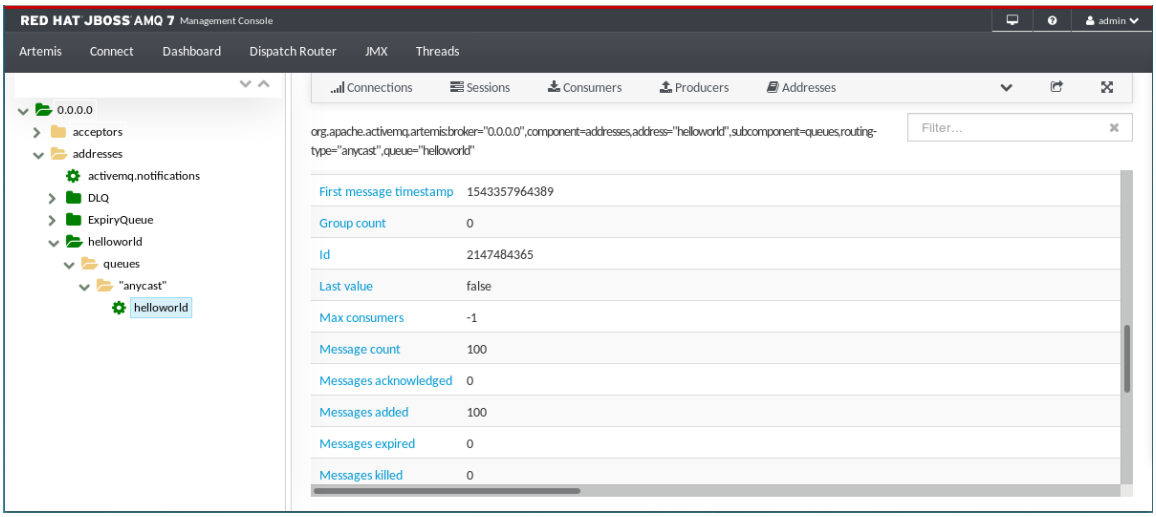
b. 브로커 인스턴스를 만들 때 생성한 기본 사용자 이름 및 기본 암호를 사용하여 콘솔에 로그인합니다.

특성 탭이 표시됩니다.

c. **Attributes** 탭에서 **address** → **helloworld** → 대기열 → **"anycast"** → **helloworld** 로 이동합니다.

이전 단계에서는 메시지를 **helloworld** 주소로 전송했습니다. 이로 인해 큐(**helloworld** 라고도 함)를 사용하여 새 **anycast helloworld**주소가 생성되었습니다. **Message count** 특성은 **helloworld** 로 전송된 **100**개의 모든 메시지가 현재 이 큐에 저장됨을 보여줍니다.

그림 **4.1.** 메시지 수



3. **artemis consumer** 명령을 사용하여 브로커에 저장된 메시지의 **50**개를 사용합니다.

이 명령은 이전에 브로커에 보낸 메시지의 **50**개를 사용합니다.

```
$ /opt/redhat/amq-broker/amq-broker-7.2.0/bin/artemis consumer --destination
helloworld --message-count 50 --url tcp://localhost:61616

Consumer:: filter = null
Consumer ActiveMQQueue[helloworld], thread=0 wait until 50 messages are
```

```
consumed
Consumer ActiveMQQueue[helloworld], thread=0 Consumed: 50 messages
Consumer ActiveMQQueue[helloworld], thread=0 Consumer thread finished
```

4. 웹 콘솔에서 **Message count** 가 이제 **50**인지 확인합니다.

   50개의 메시지가 사용되었으며, 이 경우 **helloworld** 큐에 저장된 **50**개의 메시지가 남습니다.

5. 브로커를 중지하고 나머지 **50**개의 메시지가 여전히 **helloworld** 큐에 저장되었는지 확인합니다.

   a. 브로커가 실행 중인 터미널에서 **Ctrl+C** 를 눌러 브로커를 중지합니다.

   b. 브로커를 다시 시작합니다.

   ```
   $ /var/opt/amq-broker/mybroker/bin/artemis run
   ```

   c. 웹 콘솔에서 **helloworld** 대기열로 돌아가 큐에 아직 **50**개의 메시지가 저장되어 있는지 확인합니다.

6. 나머지 **50**개의 메시지를 사용합니다.

```
$ /opt/redhat/amq-broker/amq-broker-7.2.0/bin/artemis consumer --destination
helloworld --message-count 50 --url tcp://localhost:61616

Consumer:: filter = null
Consumer ActiveMQQueue[helloworld], thread=0 wait until 50 messages are
consumed
Consumer ActiveMQQueue[helloworld], thread=0 Consumed: 50 messages
Consumer ActiveMQQueue[helloworld], thread=0 Consumer thread finished
```

7. 웹 콘솔에서 메시지 수가 **0**인지 확인합니다.

   helloworld 큐에 저장된 모든 메시지가 사용되었으며 대기열이 비어 있습니다.

4.4. 브로커 인스턴스 중지

독립 실행형 브로커를 생성하고 테스트 메시지를 생성 및 사용한 후 브로커 인스턴스를 중지할 수 있습

니다.

이 절차에서는 브로커를 수동으로 중지하므로 모든 클라이언트 연결을 강제로 종료합니다. 프로덕션 환경에서는 클라이언트 연결을 올바르게 닫을 수 있도록 브로커를 정상적으로 중지하도록 구성해야 합니다.

절차

- **artemis stop** 명령을 사용하여 브로커 인스턴스를 중지합니다.

```
$ /var/opt/amq-broker/mybroker/bin/artemis stop
2018-12-03 14:37:30,630 INFO  [org.apache.activemq.artemis.core.server] AMQ221002:
Apache ActiveMQ Artemis Message Broker version 2.6.1.amq-720004-redhat-1
[b6c244ef-f1cb-11e8-a2d7-0800271b03bd] stopped, uptime 35 minutes
Server stopped!
```

5장. AMQ BROKER 예제 실행

AMQ Broker에는 제품의 기본 및 고급 기능을 보여주는 많은 예제 프로그램이 포함되어 있습니다. AMQ Broker의 기능에 익숙해지도록 다음 예제를 실행할 수 있습니다.

AMQ Broker 예제를 실행하려면 먼저 Apache Maven 및 AMQ Maven 리포지토리를 설치 및 구성하여 시스템을 설정해야 합니다. 그런 다음 Maven을 사용하여 AMQ Broker 예제 프로그램을 실행합니다.

5.1. AMQ BROKER 예제를 실행하도록 머신 설정

포함된 AMQ Broker 예제 프로그램을 실행하려면 먼저 Maven 및 AMQ Maven 리포지토리를 다운로드하여 설치하고 Maven 설정 파일을 구성해야 합니다.

5.1.1. Maven 다운로드 및 설치

AMQ Broker 예제를 실행하려면 Maven이 필요합니다.

절차

1. Apache Maven 다운로드 페이지로 이동하여 운영 체제의 최신 배포를 다운로드합니다.
2. 운영 체제에 Maven을 설치합니다.

   자세한 내용은 Apache Maven 설치를 참조하십시오.

추가 리소스

- Maven에 대한 자세한 내용은 Apache Maven 소개를 참조하십시오.

5.1.2. AMQ Maven 리포지토리 다운로드 및 설치

Maven이 머신에 설치되면 AMQ Maven 리포지토리를 다운로드하여 설치합니다. 이 리포지토리는 Red Hat 고객 포털에서 사용할 수 있습니다.

1. 웹 브라우저에서 https://access.redhat.com/downloads/ 으로 이동하여 로그인합니다.

제품 다운로드 페이지가 표시됩니다.

2. Integration 및 Automation 섹션에서 Red Hat AMQ Broker 링크를 클릭합니다.

Software Downloads 페이지가 표시됩니다.

3. 버전 드롭다운 메뉴에서 원하는 AMQ Broker 버전을 선택합니다.

4. 릴리스 탭에서 AMQ Broker Maven 리포지토리에 대한 다운로드 링크를 클릭합니다.

AMQ Maven 리포지토리 파일은 zip 파일로 다운로드됩니다.

5. 시스템에서 AMQ 리포지토리 파일의 압축을 풉니다.

시스템에 maven-repository/ 라는 하위 디렉터리에 Maven 리포지토리가 포함된 새 디렉터리가 생성됩니다.

5.1.3. Maven 설정 파일 구성

AMQ Maven 리포지토리를 다운로드하여 설치한 후 Maven 설정 파일에 리포지토리를 추가해야 합니다.

절차

1. Maven settings.xml 파일을 엽니다.

settings.xml 파일은 일반적으로 ${user.home}/.m2/ 디렉터리에 있습니다.

- Linux의 경우 ~/.m2/입니다.
- Windows의 경우 \Documents 및 Settings\.m2\ 또는 \Users\.m2\ 입니다.

${user.home}/.m2/ 에서 settings.xml 파일을 찾을 수 없는 경우 Maven 설치의 conf/ 디렉

터리에 기본 버전이 있습니다. 기본 **settings.xml** 파일을 **${user.home}/.m2/** 디렉터리에 복사합니다.

2. **&lt ;profiles& gt;** 요소에서 **AMQ Maven** 리포지토리에 대한 프로필을 추가합니다.

```
<!-- Configure the JBoss AMQ Maven repository -->
<profile>
 <id>jboss-amq-maven-repository</id>
 <repositories>
  <repository>
   <id>jboss-amq-maven-repository</id>
   <url>file://<JBoss-AMQ-repository-path></url> (1)
   <releases>
    <enabled>true</enabled>
   </releases>
   <snapshots>
    <enabled>false</enabled>
   </snapshots>
  </repository>
 </repositories>
 <pluginRepositories>
  <pluginRepository>
   <id>jboss-amq-maven-repository</id>
   <url>file://<JBoss-AMQ-repository-path></url> (2)
   <releases>
    <enabled>true</enabled>
   </releases>
   <snapshots>
    <enabled>false</enabled>
   </snapshots>
  </pluginRepository>
 </pluginRepositories>
</profile>
```

(1) (2) **& lt;JBoss-AMQ-repository-path >**를 설치한 **Maven** 리포지토리의 위치로 바꿉니다. 일반적으로 이 위치는 **/maven-repository** 로 끝납니다. 예를 들면 다음과 같습니다.

```
<url>file:///path/to/repo/amq-broker-7.2.0-maven-repository/maven-repository</url>
```

3. **&lt ;activeProfiles& gt;** 요소에서 **AMQ Maven** 리포지토리를 **active**로 설정합니다.

```
<activeProfiles>
 <activeProfile>jboss-amq-maven-repository</activeProfile>
...
</activeProfiles>
```

4. **Maven** 설치에서 기본 **settings.xml** 을 복사한 경우 기본적으로 주석 처리된 경우 **< active-profiles >** 섹션의 주석을 제거합니다.

5. 설정 저장 및 닫기 **settings.xml.**

6. 캐시된 **${user.home}/.m2/repository/** 디렉터리를 제거합니다.

Maven 리포지토리에 오래된 아티팩트가 포함된 경우 프로젝트를 빌드하거나 배포할 때 다음과 같은 **Maven** 오류 메시지 중 하나가 발생할 수 있습니다.

- 아티팩트 **<artifact-name>** 누락
- **[ERROR]** 프로젝트 **<project-name>**에서 목표를 실행하지 못했습니다. **<project-name>**의 종속성을 해결할 수 없습니다.

5.2. AMQ BROKER 예제 프로그램

AMQ Broker에는 **AMQ Broker** 기능 및 지원되는 메시징 프로토콜을 사용하는 방법을 보여주는 **90** 개 이상의 예제 프로그램이 포함되어 있습니다.

예제 프로그램은 **< *install_dir>*; /examples** 에 있으며 다음을 포함합니다.

- 기능

  다음과 같은 브로커별 기능

  - 클러스터 **-** 로드 밸런싱 및 배포 기능을 보여주는 예
  - **HA -** 페일오버 및 재연결 기능을 보여주는 예
  - **perf -** 서버에서 몇 가지 성능 테스트를 실행할 수 있는 예

- 표준 - 다양한 브로커 기능을 보여주는 예
- 하위 모듈 - 통합된 외부 모듈의 예

- 프로토콜

  지원되는 각 메시징 프로토콜에 대한 예는 다음과 같습니다.

  - AMQP
  - MQTT
  - OpenWire
  - STOMP

추가 리소스

- 각 예제 프로그램에 대한 설명은 Apache Artemis 설명서의 예를 참조하십시오. https://activemq.apache.org/artemis/docs/latest/examples.html

5.3. AMQ BROKER 예제 프로그램 실행

AMQ Broker에는 제품의 기본 및 고급 기능을 보여주는 많은 예제 프로그램이 포함되어 있습니다. Maven을 사용하여 이러한 프로그램을 실행합니다.

사전 요구 사항

- AMQ Broker 예제를 실행하도록 머신을 설정해야 합니다.

  자세한 내용은 5.1절. “AMQ Broker 예제를 실행하도록 머신 설정”의 내용을 참조하십시오.

절차

1. 실행할 예제의 디렉터리로 이동합니다.

   예제 프로그램은 < *install_dir*> /examples 에 있습니다. 예를 들면 다음과 같습니다.

```
$ cd <install_dir>/examples/features/standard/queue
```

2. mvn clean verify 명령을 사용하여 예제 프로그램을 실행합니다.

   Maven은 브로커를 시작하고 예제 프로그램을 실행합니다. 예제 프로그램을 처음 실행할 때 Maven은 실행하는데 시간이 걸릴 수 있는 누락된 종속성을 다운로드합니다.

   이 경우 큐 예제 프로그램이 실행되어 생산자를 생성하고 테스트 메시지를 보낸 다음 메시지를 수신하는 소비자를 생성합니다.

```
$ mvn clean verify
[INFO] Scanning for projects...
[INFO]
[INFO] ------------< org.apache.activemq.examples.broker:queue >--------------
[INFO] Building ActiveMQ Artemis JMS Queue Example 2.6.1.amq-720004-redhat-1
[INFO] --------------------------------[ jar ]---------------------------------
...
server-out:2018-12-05 16:37:57,023 INFO  [org.apache.activemq.artemis.core.server]
AMQ221001: Apache ActiveMQ Artemis Message Broker version 2.6.1.amq-720004-
redhat-1 [0.0.0.0, nodeID=06f529d3-f8d6-11e8-9bea-0800271b03bd]
[INFO] Server started
[INFO]
[INFO] --- artemis-maven-plugin:2.6.1.amq-720004-redhat-1:runClient (runClient) @
queue ---
Sent message: This is a text message
Received message: This is a text message
[INFO]
[INFO] --- artemis-maven-plugin:2.6.1.amq-720004-redhat-1:cli (stop) @ queue ---
server-out:2018-12-05 16:37:59,519 INFO  [org.apache.activemq.artemis.core.server]
AMQ221002: Apache ActiveMQ Artemis Message Broker version 2.6.1.amq-720004-
redhat-1 [06f529d3-f8d6-11e8-9bea-0800271b03bd] stopped, uptime 3.734 seconds
server-out:Server stopped!
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  48.681 s
[INFO] Finished at: 2018-12-05T16:37:59-05:00
[INFO] ------------------------------------------------------------------------
```

참고

일부 예제 프로그램은 **UDP** 클러스터링을 사용하며 기본적으로 사용자 환경에서 작동하지 않을 수 있습니다. 이 예제를 성공적으로 실행하려면 **224.0.0.0**으로 전송된 트래픽을 루프백 인터페이스로 리디렉션합니다.

```
$ sudo route add -net 224.0.0.0 netmask 240.0.0.0 dev lo
```

6장. 다음 단계

AMQ Broker를 설치하고 기본 구성 설정으로 독립 실행형 브로커를 생성한 후 메시징 요구 사항을 충족하고 메시징 클라이언트 애플리케이션을 연결 및 모니터링 및 관리할 수 있습니다. 이러한 목표를 달성하는 데 도움이 되는 추가 리소스를 사용할 수 있습니다.

브로커 구성

AMQ Broker 구성을 사용하여 요구 사항에 맞게 브로커를 구성합니다. 다음을 구성할 수 있습니다.

- 클라이언트 연결을 수락할 브로커
- 주소 공간 (**point-to-point** 및 **publish-subscribe** 메시징 포함)
- 메시지 지속성
- 브로커 리소스 사용(리소스 제한, 메시지 페이징 및 대규모 메시지 지원 포함)
- 중복 메시지 탐지
- 로깅

브로커 보안

Configuring AMQ Broker 를 사용하여 브로커를 보호하는 다음 방법을 구현합니다.

- 게스트/익명 액세스 제어
- 기본 사용자 및 암호 액세스 제어
- 인증서 기반 액세스 제어

- LDAP 통합
- Kerberos 통합

클러스터링 및 고가용성 설정

AMQ Broker 구성을 사용하여 브로커 클러스터를 구성하고 메시징 처리량을 늘리기 위해 추가 브로커를 추가합니다. 메시지 안정성을 높이기 위해 고가용성을 구성할 수도 있습니다.

메시징 클라이언트 애플리케이션 생성

AMQ Clients 개요 를 사용하여 AMQ Clients에 대해 알아보고 브로커에 연결하고 메시지를 생성하고 사용하는 메시징 클라이언트 애플리케이션을 생성하는 데 도움이 되는 방법을 알아보십시오.

브로커 모니터링 및 관리

AMQ Broker 를 관리하여 실행 후 브로커(또는 브로커)를 모니터링하고 관리합니다.

부록 A. 서브스크립션 사용

AMQ는 소프트웨어 서브스크립션을 통해 제공됩니다. 서브스크립션을 관리하려면 Red Hat 고객 포털에서 계정에 액세스하십시오.

A.1. 귀하의 계정에 액세스

절차

1. access.redhat.com 으로 이동합니다.
2. 아직 계정이 없는 경우 계정을 생성합니다.
3. 계정에 로그인합니다.

A.2. 서브스크립션 활성화

절차

1. access.redhat.com 으로 이동합니다.
2. 내 서브스크립션으로 이동합니다.
3. 서브스크립션을 활성화하여 16자리 활성화 번호를 입력합니다.

A.3. 릴리스 파일 다운로드

.zip, .tar.gz 및 기타 릴리스 파일에 액세스하려면 고객 포털을 사용하여 다운로드할 관련 파일을 찾습니다. RPM 패키지 또는 Red Hat Maven 리포지토리를 사용하는 경우에는 이 단계가 필요하지 않습니다.

절차

1. 브라우저를 열고 access.redhat.com/downloads 에서 Red Hat 고객 포털 제품 다운로드 페이지에 로그인합니다.

2. **INTEGRATION AND AUTOMATION** 카테고리에서 **Red Hat AMQ** 항목을 찾습니다.

3. 원하는 **AMQ** 제품을 선택합니다. **Software Download** 페이지가 열립니다.

4. 구성 요소에 대한 다운로드 링크를 클릭합니다.

A.4. 패키지용 시스템 등록

Red Hat Enterprise Linux에 이 제품의 **RPM** 패키지를 설치하려면 시스템을 등록해야 합니다. 다운로드한 릴리스 파일을 사용하는 경우 이 단계는 필요하지 않습니다.

절차

1. access.redhat.com 으로 이동합니다.

2. **Registration Assistant** 로 이동합니다.

3. **OS** 버전을 선택하고 다음 페이지로 이동합니다.

4. 시스템 터미널에서 나열된 명령을 사용하여 등록을 완료합니다.

시스템 등록에 대한 자세한 내용은 다음 리소스 중 하나를 참조하십시오.

- **Red Hat Enterprise Linux 7 -** 시스템 등록 및 서브스크립션 관리
- **Red Hat Enterprise Linux 8 -** 시스템 등록 및 서브스크립션 관리

부록 B. RED HAT MAVEN 리포지토리 사용

이 섹션에서는 소프트웨어에서 Red Hat 제공 Maven 리포지토리를 사용하는 방법에 대해 설명합니다.

B.1. 온라인 리포지토리 사용

Red Hat은 Maven 기반 프로젝트와 함께 사용할 중앙 Maven 리포지토리를 유지 관리합니다. 자세한 내용은 리포지토리 시작 페이지를 참조하십시오.

Maven 설정에 리포지토리를 추가하면 POM 파일이 리포지토리 구성을 재정의하지 않고 포함된 프로필이 활성화된 경우 해당 구성이 사용자가 소유한 모든 Maven 프로젝트에 적용됩니다.

절차

1. Maven settings.xml 파일을 찾습니다. 일반적으로 사용자 홈 디렉터리의 .m2 디렉토리 내에 있습니다. 파일이 없는 경우 텍스트 편집기를 사용하여 생성합니다.

   Linux 또는 UNIX에서:

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

   Windows에서 다음을 수행합니다.

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

2. 다음 예제와 같이 Red Hat 리포지토리를 포함하는 새 프로필을 settings.xml 파일의 profiles 요소에 추가합니다.

   예: Red Hat 리포지토리가 포함된 Maven settings.xml 파일

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

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

Maven 구성에 대한 자세한 내용은 **Maven** 설정 참조 를 참조하십시오.

B.2. 로컬 리포지터리 사용

Red Hat은 일부 구성 요소에 파일 기반 **Maven** 리포지토리를 제공합니다. 이러한 파일은 로컬 파일 시스템으로 추출할 수 있는 다운로드 가능한 아카이브로 제공됩니다.

로컬 추출된 리포지토리를 사용하도록 **Maven**을 구성하려면 **Maven** 설정 또는 **POM** 파일에 다음 **XML**을 적용합니다.

```
<repository>
  <id>red-hat-local</id>
  <url>${repository-url}</url>
  <releases>
    <enabled>true</enabled>
  </releases>
  <snapshots>
    <enabled>false</enabled>
```

```
  </snapshots>
</repository>
<pluginRepository>
  <id>red-hat-local</id>
  <url>${repository-url}</url>
  <releases>
    <enabled>true</enabled>
  </releases>
  <snapshots>
    <enabled>false</enabled>
  </snapshots>
</pluginRepository>
```

${repository-url} 은 추출된 리포지토리의 로컬 파일 시스템 경로가 포함된 파일 **URL**이어야 합니다.

표 B.1. 로컬 Maven 리포지토리의 URL 예

운영 체제	파일 시스템 경로	URL
Linux 또는 UNIX	**/home/alice/maven-repository**	**file:/home/alice/maven-repository**
Windows	**C:\repos\red-hat**	**file:C:\repos\red-hat**

2024-06-11에 최종 업데이트된 문서