Red Hat Fuse 7.11

마이그레이션 가이드

Red Hat Fuse 7.11로 마이그레이션

Last Updated: 2023-07-25

Red Hat Fuse 7.11 마이그레이션 가이드

Red Hat Fuse 7.11로 마이그레이션

법적 공지

초록

이 가이드를 사용하여 Fuse 설치를 최신 Red Hat Fuse 버전으로 업그레이드할 때 도움이 됩니다.

차례

머리말

이 가이드에서는 Red Hat Fuse 및 Fuse 애플리케이션 업데이트에 대한 정보를 제공합니다.

참고

이 가이드의 지침에 따라 Fuse 6에서 최신 Fuse 7 릴리스로 마이그레이션하려면 Red Hat Fuse 7.0 마이그레이션 가이드 의 지침을 따라야 합니다.

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 에서 참조하십시오.

1장. OPENSHIFT에서 FUSE 업그레이드

Fuse 7.11에는 OCP(OpenShift Container Platform) 4.9 이상에서 작업할 수 있는 업데이트가 포함되어 있습니다. OCP 4.10으로 업그레이드하려는 경우 OCP를 버전 4.10으로 업그레이드하기 **전에** Fuse를 버전 7.11로 업그레이드해야 합니다. 이전 버전의 Fuse(7.10)에서는 OCP 4.9 이상을 지원하지 않습니다.

자세한 내용은 *Fuse on OpenShift Guide* 를 참조하십시오.

2장. FUSE ONLINE 업그레이드

경우에 따라 패치 및 보안 수정 사항을 통합하는 새로운 애플리케이션 이미지가 Fuse용으로 릴리스됩니다. Red Hat의 에라타 업데이트 채널을 통해 이러한 업데이트에 대한 알림을 받습니다. 그런 다음 Fuse 이미지를 업그레이드할 수 있습니다.

OCP 4.x의 경우 OperatorHub(OCP 4.x)를 사용하여 Upgrading Fuse의 단계에 따라 OpenShift OperatorHub를 사용하여 Fuse 7.10에서 7.11로 업그레이드합니다.

Fuse 7.11으로 업그레이드하려면 기존 통합을 변경해야 하는지 확인해야 합니다. 변경 사항이 필요하지 않은 경우에도 Fuse를 업그레이드할 때 실행 중인 통합을 다시 게시해야 합니다.

OperatorHub를 사용하여 Fuse 업그레이드 (OCP 4.x)

OpenShift OperatorHub를 사용하여 Fuse Online 7.10에서 7.11로 업그레이드합니다.

1. Fuse 7.9.x에서 Fuse Online 7.10.1로 업그레이드하려면 먼저 Fuse **Online 7.9.x에서 7.10.1로 업그레이드하려면 수동 업그레이드 단계 릴리스 노트에 설명된 대로 Fuse 7.10.0으로 수동으로 업그레이드해야** 합니다.

2. Fuse 7.11에는 OCP(OpenShift Container Platform) 4.6 이상이 필요합니다. OCP 4.5 이하를 사용하는 경우 Fuse 7.11로 업그레이드하려면 OCP 4.6 이상으로 업그레이드해야 합니다.

3. OCP 4.9에서 7.11로 업그레이드할 때 Fuse Operator 업그레이드 프로세스 중에 다음 경고가 표시됩니다.
   W1219 18:38:58.064578 1 warnings.go:70] extensions/v1beta1 Ingress는 v1.14 이상에서 더 이상 사용되지 않으며 v1.22 이상에서 사용할 수 없습니다. networking.k8s.io/v1 Ingress를 사용합니다.

   클라이언트(Fuse를 Kubernetes/OpenShift API 초기화 코드에 사용하는) 클라이언트가 더 이상 사용되지 않는 Ingress 버전에 액세스하므로 이 경고가 표시됩니다. 이 경고는 더 이상 사용되지 않는 API를 완전히 사용할 수 있는 지표가 아니며 Fuse 7.11으로 업그레이드하는 데 문제가 없습니다.

Fuse Online 7.10 또는 이전 버전의 업그레이드 프로세스는 Fuse Online을 설치할 때 **선택한 승인** 전략에 따라 다릅니다.

- **자동** 업데이트의 경우 새 버전의 Fuse Operator를 사용할 수 있는 경우 OpenShift Operator Lifecycle Manager(OLM)는 사용자의 개입 없이 Fuse Online의 실행 중인 인스턴스를 자동으로 업그레이드합니다.

- **수동** 업데이트의 경우 최신 버전의 Operator를 사용할 수 있으면 OLM에서 업데이트 요청을 생성합니다. 그런 다음 클러스터 관리자는 OpenShift 문서의 **보류 중인 Operator 업그레이드 섹션에 설명된 대로 Fuse Online Operator** 가 새 버전으로 업데이트되도록 해당 업데이트 요청을 수동으로 승인해야 합니다.

인프라 업그레이드 중 및 이후에 기존 통합은 이전 버전의 Fuse 라이브러리 및 종속 항목을 계속 실행합니다.

업데이트된 Fuse Online 버전과 함께 기존 통합을 실행하려면 통합을 다시 게시해야 합니다.

3장. SPRING BOOT 2로 업그레이드

이 장에서는 Spring Boot 1에서 애플리케이션을 Spring Boot 2.0으로 업그레이드하는 방법을 설명합니다.

3.1. 사전 준비 사항

Spring Boot 2로의 마이그레이션을 시작하기 전에 시스템 요구 사항 및 종속 항목을 검토해야 합니다.

- 최신 1.5.x 버전으로 업그레이드
  - 시작하기 전에 최신 1.5.x 사용 가능한 버전으로 업그레이드하십시오. 이는 해당 라인의 최신 종속 항목에 대해 빌드하도록 하기 위한 것입니다.
- 종속 항목 검토
  - Spring Boot 2로 마이그레이션하면 여러 종속 항목을 업그레이드합니다. 2.0.x에 대한 종속성 관리로 1.5.x의 종속성 관리를 검토하여 프로젝트의 영향을 받는 방법을 평가합니다.
  - Spring Boot에서 관리하지 않는 종속 항목에 대한 호환 가능한 버전을 확인한 다음 이에 대한 명시적 버전을 정의합니다.
- 사용자 정의 구성 검토
  - 프로젝트에서 정의하는 사용자 정의 구성은 업그레이드 시 검토해야 할 수 있습니다. 표준 자동 구성을 사용하여 이 값을 교체할 수 있는 경우 업그레이드하기 전에 수행합니다.
- 시스템 요구 사항 검토
  - Spring Boot 2.0에는 Java 8 이상이 필요합니다.
  - 또한 Spring Framework 5.0이 필요합니다.
  - Java 6 및 7은 더 이상 지원되지 않습니다.

3.2. SPRING BOOT 1에서 SPRING BOOT 2로 업그레이드

프로젝트 상태 및 해당 종속 항목을 검토한 후 Spring Boot 2.x의 최신 유지 관리 릴리스로 업그레이드하십시오. 단계에서 업그레이드하는 것이 좋습니다. 예를 들어 먼저 Spring Boot 1.5에서 Spring Boot 2.0으로 업그레이드한 다음 2.1으로 업그레이드한 다음 Spring Boot 2의 최신 유지 관리 릴리스로 업그레이드합니다.

구성 속성 마이그레이션

Spring Boot 2.0을 사용하면 많은 구성 속성의 이름이 변경되거나 제거되었습니다. 따라서 **application.properties/application.yml** 을 적절하게 업데이트해야 합니다. 이를 통해 새로운 **spring-boot-properties-migrator** 모듈의 도움을 받을 수 있습니다. 프로젝트에 종속성으로 추가되면 애플리케이션의 환경을 분석하고 시작 시 진단을 출력할 뿐만 아니라 런타임 시 속성도 일시적으로 마이그레이션합니다.

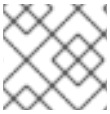
절차

1. 프로젝트 **pom.xml** 의 종속성 섹션에 **spring-boot-properties-migrator** 모듈을 추가합니다.

```
<dependency>
 <groupId>org.springframework.boot</groupId>
 <artifactId>spring-boot-properties-migrator</artifactId>
```

```
 <scope>runtime</scope>
</dependency>

runtime("org.springframework.boot:spring-boot-properties-migrator")
```



참고

마이그레이션이 완료되면 이 모듈을 프로젝트의 종속 항목에서 제거하십시오.

4장. FABRIC8 MAVEN 플러그인에서 OPENSHIFT MAVEN 플러그인으로 마이그레이션

fabric8-maven-plugin 은 Fuse 7.11에서 완전히 제거되었습니다. OpenShift에서 Maven 프로젝트를 빌드하고 배포하는 대신 **openshift-maven-plugin** 을 사용하는 것이 좋습니다.

절차

다음 지침을 사용하여 openshift-maven 플러그인을 사용할 수 있도록 애플리케이션을 업데이트합니다.

1. 애플리케이션의 **src/main/fabric8** 디렉터리의 이름을 **src/main/jkube** 로 변경합니다.
2. 프로젝트의 pom.xml에서 **org.jboss.redhat-fuse:fabric8-maven-plugin** 종속성을 찾아 **org.jboss.redhat.redhat-fuse:openshift-maven-plugin**. 샘플 pom.xml 을 참조하십시오.
3. 종속 항목을 확인합니다. 예를 들어 **org.arquillian.cube:arquillian-cube-openshift,org.jboss.arquillian.junit:arquillian-junit-container,io.fabric8:kubernetes-assertions** 는 예제에서는 더 이상 사용되지 않으며 더 이상 필요하지 않을 수 있습니다.
4. 마이그레이션 후 API 변경 사항을 반영하는 데 사용할 수 있는 몇 가지 샘플 테스트를 생성할 수 있습니다. 자세한 내용은 Spring Boot Camel 빠른 시작 의 샘플 테스트를 참조하십시오.

추가 리소스

- OpenShift Maven 플러그인.

5장. SPRING BOOT 독립형에서 FUSE 애플리케이션 업그레이드

Spring Boot에서 Fuse 애플리케이션을 업그레이드하려면 다음을 수행합니다.

- 5.1절. "Camel 마이그레이션 고려 사항" 에 설명된 대로 Apache Camel 업데이트를 고려해야 합니다.
- 올바른 Fuse 버전을 사용하고 있는지 확인하려면 Fuse 프로젝트의 Maven 종속성을 업데이트해야 합니다.

일반적으로 Maven을 사용하여 Fuse 애플리케이션을 빌드합니다. Maven은 Apache의 무료 오픈 소스 빌드 툴입니다. Maven 구성은 Fuse 애플리케이션 프로젝트의 **pom.xml** 파일에 정의되어 있습니다. Fuse 프로젝트를 빌드하는 동안 기본 동작은 Maven이 외부 리포지토리를 검색하고 필요한 아티팩트를 다운로드하는 것입니다. Maven 빌드 프로세스에서 올바른 Fuse 지원 아티팩트 세트를 선택할 수 있도록 BOM(Fuse of material)에 대한 종속성을 **pom.xml** 파일에 추가합니다.

다음 섹션에서는 Maven 종속 항목에 대한 정보와 Fuse 프로젝트에서 업데이트하는 방법을 설명합니다.

- 5.2절. "Maven 종속 항목 정보"
- 5.3절. "Fuse 프로젝트의 Maven 종속성 업데이트"

5.1. CAMEL 마이그레이션 고려 사항

MongoClients 팩토리를 사용하여 MongoDB에 연결 생성

Fuse 7.11에서 **com.mongodb.MongoClient** 대신 **com.mongodb.client.MongoClient** 를 사용하여 MongoDB에 대한 연결을 만듭니다(전체 경로에 추가 *.client* 하위 패키지 참조).

기존 Fuse 애플리케이션이 **camel-mongodb** 구성 요소를 사용하는 경우 다음을 수행해야 합니다.

- 애플리케이션을 업데이트하여 연결 빈을 **com.mongodb.client.MongoClient** 인스턴스로 만듭니다.
  예를 들어 다음과 같이 MongoDB에 대한 연결을 생성합니다.

  ```
  import com.mongodb.client.MongoClient;
  ```

  그런 다음 다음 예와 같이 MongoClient 8080을 생성할 수 있습니다.

  ```
  return MongoClients.create("mongodb://admin:password@192.168.99.102:32553");
  ```

- 평가하고 필요한 경우 MongoClient 클래스에서 노출하는 메서드와 관련된 모든 코드를 리팩터링합니다.

Camel 2.23

Red Hat Fuse는 Apache Camel 2.23을 사용합니다. Fuse 7.8로 업그레이드할 때 Camel 2.22 및 2.23에 대한 다음 업데이트를 고려해야 합니다.

Camel 2.22 업데이트

- Camel이 Spring Boot v1에서 v2로 업그레이드되었으므로 v1은 더 이상 지원되지 않습니다.
- Spring Framework 5로 업그레이드 Camel은 Spring 4.3.x와도 함께 작동해야 하지만 향후 Spring 5.x는 향후 릴리스에서 최소 Spring 버전이 될 것입니다.

- Karaf 4.2로 업그레이드했습니다. Camel을 Karaf 4.1에서 실행할 수 있지만 이 릴리스에서 Karaf 4.2만 공식적으로 지원합니다.
- toD DSL을 사용하여 최적화하여 가능한 구성 요소에 대해 끝점 및 생산자를 재사용합니다. 예를 들어 HTTP 기반 구성 요소는 동일한 호스트로 전송되는 동적 URI와 함께 생산자(HTTP 클라이언트)를 재사용합니다.
- 이제 read-lock idempotent/idempotent-changed가 있는 File2 소비자는 이제 파일이 in-process로 간주될 때 릴리스 작업을 지연하도록 구성할 수 있습니다. 이 파일은 프로세스 내/활성 클러스터 설정에서 사용 가능하며, 다른 노드에서 처리된 파일을 처리할 수 있는 파일로 너무 빠르게 표시되지 않도록 합니다(RemoveOnCommit=true).
- 요청/응답 모드에서 Netty4 프로듀서에서 사용자 정의 요청/일반 상관 ID 관리자 구현을 플러그인할 수 있습니다. 이제 Twitter 구성 요소에서 확장 모드를 사용하여 140자를 초과한 event를 지원합니다.
- rest DSL 생산자는 이제 endpointProperties를 사용하여 REST 구성에서 구성할 수 있습니다.
- Kafka 구성 요소에서는 Camel과 Kafka 메시지 간의 헤더 매핑을 제어하기 위한 사용자 지정 구현을 플러그인하도록 HeaderFilterStrategy를 지원합니다.
- REST DSL은 이제 클라이언트 요청 검증을 지원하여 REST 서비스에 Content-Type/Accept 헤더가 가능한지 확인합니다.
- Camel에는 Camel 구현 또는 Spring Cloud를 사용하여 서비스 레지스트리(예: consul, etcd 또는 Zookeeper)에 경로를 등록할 수 있는 Service Registry SPI가 있습니다.
- SEDA 구성 요소의 기본 대기열 크기는 무제한이 아닌 1000입니다.
- 다음과 같은 중요한 문제가 해결되었습니다.
  - CXF 연속 시간제한 문제가 camel-cxf 소비자와 관련된 수정으로 인해 소비자가 시간 초과를 트리거하는 대신 데이터가 포함된 응답을 반환하도록 할 수 있습니다.
  - 고정 camel-cxf 소비자는 강력한 단방향 작업을 사용할 때 UoW를 릴리스하지 않습니다.
  - AdviceWith를 사용하여 수정되었으며 onException에서 weave 메서드를 사용하는 경우 작동하지 않습니다.
  - 병렬 처리 및 스트리밍 모드에서 고정 Splitter는 메시지 본문을 차단하는 동안 처음으로 호출된 next() 메서드 호출에서 예외를 발생시킬 때 차단될 수 있습니다.
  - autoCommitEnable=false인 경우 Kafka 소비자가 자동 커밋되지 않도록 수정했습니다.
  - 고정 파일 소비자는 기본적으로 markerFile을 읽기 잠금으로 사용했으며 이는 none이어야 했습니다.
  - Kafka와의 수동 커밋을 사용하여 이전 레코드 오프셋이 아닌 현재 레코드 오프셋을 제공합니다(및 먼저 -1).
  - Java DSL의 고정 콘텐츠 기반 라우터는 서술자가 있는 경우 속성 자리 표시자를 확인하지 않을 수 있습니다.

Camel 2.23 업데이트

- Spring Boot 2.1으로 업그레이드했습니다.

- 이제 Spring-boot 자동 구성을 사용하여 추가 구성 요소 수준 옵션을 구성할 수 있습니다. 이러한 옵션은 Spring-boot 구성 요소 메타데이터 JSON 파일 설명자에 포함되어 있습니다.
- 모든 구성 요소, 데이터 형식 및 언어에 대한 모든 Spring Boot 자동 구성 옵션이 포함된 문서 섹션을 추가했습니다.
- 모든 Camel Spring Boot Starter JAR에는 Spring Boot 자동 구성을 최적화하기 위해 JAR에 **META-INF/spring-autoconfigure-metadata.properties** 파일이 포함되어 있습니다.
- Throttler는 동적 표현식을 기반으로 한 상관관계 그룹을 지원하여 다양한 제한 세트로 메시지를 그룹화할 수 있습니다.
- Hystrix EIP는 이제 Camel의 오류 처리기에 대한 상속을 허용하므로 재전송을 통해 오류 처리를 활성화한 경우 전체 Hystrix EIP 블록을 다시 시도할 수 있습니다.
- SQL 및 ElSql 소비자는 이제 경로 양식에서 동적 쿼리 매개변수를 지원합니다. 이 기능은 간단한 표현식을 사용하여 빈을 호출하도록 제한됩니다.
- swagger-restdsl maven 플러그인은 이제 Swagger 사양 파일에서 DTO 모델 클래스 생성을 지원합니다.
- 다음과 같은 중요한 문제가 해결되었습니다.
  - Aggregator2는 모든 그룹의 강제 완료를 위해 제어 헤더를 전파하지 않도록 수정되었으므로 라우팅 중에 다른 집계기 EIP가 나중에 사용 중인 경우 다시 발생하지 않습니다.
  - 오류 처리기에서 재전송이 실행된 경우 고정 추적기가 작동하지 않습니다.
  - XML 문서에 대한 기본 제공 유형 변환기는 구문 분석 오류를 stdout에 출력할 수 있으며 로깅 API를 사용하여 출력하도록 수정되었습니다.
  - 메시지 본문이 스트리밍 기반인 경우 charset 옵션을 사용하여 파일을 작성하여 파일을 작성하는 것이 작동하지 않습니다.
  - 여러 경로를 통해 라우팅할 때 재사용되지 않도록 Zipkin 루트 ID를 수정하여 단일 상위 범위로 그룹화합니다.
  - HTTP 끝점을 사용할 때 숫자가 있는 IP 주소가 포함된 경우 버그가 있었습니다.
  - 임시 대기열을 통해 요청/거부하고 수동 승인 모드를 사용하는 RabbitMQ의 문제를 해결했습니다. 임시 큐(요청/응답 가능)를 인식하지 못합니다.
  - Allow 헤더 for OPTIONS 요청(예: rest-dsl 사용 시)에서 허용되는 모든 HTTP 동사를 반환하지 않을 수 있는 다양한 HTTP 소비자 구성 요소가 수정되었습니다.
  - FluentProducerTemplate에서 thread- Cryostat 문제를 해결했습니다.

5.2. MAVEN 종속 항목 정보

BOM(Maven bill of materials) 파일의 목적은 잘 작동하는 Maven 종속성 버전 집합을 제공하여 모든 Maven 아티팩트에 대해 버전을 개별적으로 정의할 필요가 없도록 하는 것입니다.

Fuse가 실행되는 각 컨테이너에 전용 BOM 파일이 있습니다.

참고

이러한 BOM 파일을 여기에서 찾을 수 있습니다. https://github.com/jboss-fuse/redhat-fuse. 또는 BOM 파일 업데이트에 대한 정보는 최신 릴리스 노트 로 이동합니다.

Fuse BOM은 다음과 같은 이점을 제공합니다.

- **pom.xml** 파일에 종속성을 추가할 때 버전을 지정할 필요가 없도록 Maven 종속 항목에 대한 버전을 정의합니다.
- 특정 버전의 Fuse에서 완전히 테스트 및 지원되는 선별된 종속성 세트를 정의합니다.
- Fuse 업그레이드 간소화.

중요

Fuse BOM에서 정의한 종속성 세트만 Red Hat에서 지원합니다.

5.3. FUSE 프로젝트의 MAVEN 종속성 업데이트

Spring Boot 용 Fuse 애플리케이션을 업그레이드하려면 프로젝트의 Maven 종속성을 업데이트합니다.

절차

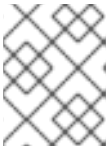
1. 프로젝트의 **pom.xml** 파일을 엽니다.
2. 다음 예와 같이 프로젝트의 **pom.xml** 파일(또는 상위 **pom.xml** 파일)에 **dependencyManagement** 요소를 추가합니다.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<project ...>
  ...
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

    <!-- configure the versions you want to use here -->
    <fuse.version>7.11.1.fuse-sb2-7_11_1-00022-redhat-00002</fuse.version>

  </properties>

  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>org.jboss.redhat-fuse</groupId>
        <artifactId>fuse-springboot-bom</artifactId>
        <version>${fuse.version}</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
  ...
</project>
```



참고

Spring Boot 버전도 업데이트해야 합니다. 일반적으로 **pom.xml** 파일의 Fuse 버전에서 찾을 수 있습니다.

```
<properties>
 <!-- configure the versions you want to use here -->
 <fuse.version>7.11.1.fuse-sb2-7_11_1-00022-redhat-00002</fuse.version>
 <spring-boot.version>2.5.13.RELEASE</spring-boot.version>
</properties>
```

3. **pom.xml** 파일을 저장합니다.

pom.xml 파일에서 BOM을 종속성으로 지정하면 아티팩트 버전을 지정하지 *않고* Maven 종속성을 **pom.xml** 파일에 추가할 수 있습니다. 예를 들어 **camel-velocity** 구성 요소에 대한 종속성을 추가하려면 **pom.xml** 파일의 **종속** 항목에 다음 XML 조각을 추가합니다.

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-velocity</artifactId>
  <scope>provided</scope>
</dependency>
```

이 종속성 정의에서 **version** 요소를 생략하는 방법을 참조하십시오.

6장. JBOSS EAP 독립 실행형에서 FUSE 애플리케이션 업그레이드

JBoss EAP에서 Fuse 애플리케이션을 업그레이드하려면 다음을 수행합니다.

- 6.1절. "Camel 마이그레이션 고려 사항" 에 설명된 대로 Apache Camel 업데이트를 고려해야 합니다.
- 올바른 Fuse 버전을 사용하고 있는지 확인하려면 Fuse 프로젝트의 Maven 종속성을 업데이트해야 합니다.
  일반적으로 Maven을 사용하여 Fuse 애플리케이션을 빌드합니다. Maven은 Apache의 무료 오픈 소스 빌드 툴입니다. Maven 구성은 Fuse 애플리케이션 프로젝트의 **pom.xml** 파일에 정의되어 있습니다. Fuse 프로젝트를 빌드하는 동안 기본 동작은 Maven이 외부 리포지토리를 검색하고 필요한 아티팩트를 다운로드하는 것입니다. Maven 빌드 프로세스에서 올바른 Fuse 지원 아티팩트 세트를 선택할 수 있도록BOM(Fuse of material)에 대한 종속성을 **pom.xml** 파일에 추가합니다.

  다음 섹션에서는 Maven 종속 항목에 대한 정보와 Fuse 프로젝트에서 업데이트하는 방법을 설명합니다.

  - 6.2절. "Maven 종속 항목 정보"
  - 6.3절. "Fuse 프로젝트의 Maven 종속성 업데이트"
- 6.4절. "Java EE 종속 항목 업그레이드" 에 설명된 대로 업그레이드된 Java EE 종속 항목을 사용하고 있는지 확인하려면 Fuse 프로젝트의 Maven 종속성을 업데이트해야 합니다.

6.1. CAMEL 마이그레이션 고려 사항

MongoClients 팩토리를 사용하여 MongoDB에 연결 생성

Fuse 7.11에서 **com.mongodb.MongoClient** 대신 **com.mongodb.client.MongoClient** 를 사용하여 MongoDB에 대한 연결을 만듭니다(전체 경로에 추가 *.client* 하위 패키지 참조).

기존 Fuse 애플리케이션이 **camel-mongodb** 구성 요소를 사용하는 경우 다음을 수행해야 합니다.

- 애플리케이션을 업데이트하여 연결 빈을 **com.mongodb.client.MongoClient** 인스턴스로 만듭니다.
  예를 들어 다음과 같이 MongoDB에 대한 연결을 생성합니다.

  ```
  import com.mongodb.client.MongoClient;
  ```

  그런 다음 다음 예와 같이 MongoClient 8080을 생성할 수 있습니다.

  ```
  return MongoClients.create("mongodb://admin:password@192.168.99.102:32553");
  ```

- 평가하고 필요한 경우 MongoClient 클래스에서 노출하는 메서드와 관련된 모든 코드를 리팩터링합니다.

Camel 2.23

Red Hat Fuse는 Apache Camel 2.23을 사용합니다. Fuse 7.8로 업그레이드할 때 Camel 2.22 및 2.23에 대한 다음 업데이트를 고려해야 합니다.

Camel 2.22 업데이트

- Camel이 Spring Boot v1에서 v2로 업그레이드되었으므로 v1은 더 이상 지원되지 않습니다.

- Spring Framework 5로 업그레이드 Camel은 Spring 4.3.x와도 함께 작동해야 하지만 향후 Spring 5.x는 향후 릴리스에서 최소 Spring 버전이 될 것입니다.
- Karaf 4.2로 업그레이드했습니다. Camel을 Karaf 4.1에서 실행할 수 있지만 이 릴리스에서 Karaf 4.2만 공식적으로 지원합니다.
- toD DSL을 사용하여 최적화하여 가능한 구성 요소에 대해 끝점 및 생산자를 재사용합니다. 예를 들어 HTTP 기반 구성 요소는 동일한 호스트로 전송되는 동적 URI와 함께 생산자(HTTP 클라이언트)를 재사용합니다.
- 이제 read-lock idempotent/idempotent-changed가 있는 File2 소비자는 이제 파일이 in-process로 간주될 때 릴리스 작업을 지연하도록 구성할 수 있습니다. 이 파일은 프로세스 내/활성 클러스터 설정에서 사용 가능하며, 다른 노드에서 처리된 파일을 처리할 수 있는 파일로 너무 빠르게 표시되지 않도록 합니다(RemoveOnCommit=true).
- 요청/응답 모드에서 Netty4 프로듀서에서 사용자 정의 요청/일반 상관 ID 관리자 구현을 플러그인할 수 있습니다. 이제 Twitter 구성 요소에서 확장 모드를 사용하여 140자를 초과한 event를 지원합니다.
- rest DSL 생산자는 이제 endpointProperties를 사용하여 REST 구성에서 구성할 수 있습니다.
- Kafka 구성 요소에서는 Camel과 Kafka 메시지 간의 헤더 매핑을 제어하기 위한 사용자 지정 구현을 플러그인하도록 HeaderFilterStrategy를 지원합니다.
- REST DSL은 이제 클라이언트 요청 검증을 지원하여 REST 서비스에 Content-Type/Accept 헤더가 가능한지 확인합니다.
- Camel에는 Camel 구현 또는 Spring Cloud를 사용하여 서비스 레지스트리(예: consul, etcd 또는 Zookeeper)에 경로를 등록할 수 있는 Service Registry SPI가 있습니다.
- SEDA 구성 요소의 기본 대기열 크기는 무제한이 아닌 1000입니다.
- 다음과 같은 중요한 문제가 해결되었습니다.
  - CXF 연속 시간제한 문제가 camel-cxf 소비자와 관련된 수정으로 인해 소비자가 시간 초과를 트리거하는 대신 데이터가 포함된 응답을 반환하도록 할 수 있습니다.
  - 고정 camel-cxf 소비자는 강력한 단방향 작업을 사용할 때 UoW를 릴리스하지 않습니다.
  - AdviceWith를 사용하여 수정되었으며 onException에서 weave 메서드를 사용하는 경우 작동하지 않습니다.
  - 병렬 처리 및 스트리밍 모드에서 고정 Splitter는 메시지 본문을 차단하는 동안 처음으로 호출된 next() 메서드 호출에서 예외를 발생시킬 때 차단될 수 있습니다.
  - autoCommitEnable=false인 경우 Kafka 소비자가 자동 커밋되지 않도록 수정했습니다.
  - 고정 파일 소비자는 기본적으로 markerFile을 읽기 잠금으로 사용했으며 이는 none이어야 했습니다.
  - Kafka와의 수동 커밋을 사용하여 이전 레코드 오프셋이 아닌 현재 레코드 오프셋을 제공합니다(및 먼저 -1).
  - Java DSL의 고정 콘텐츠 기반 라우터는 서술자가 있는 경우 속성 자리 표시자를 확인하지 않을 수 있습니다.

Camel 2.23 업데이트

- Spring Boot 2.1으로 업그레이드했습니다.
- 이제 Spring-boot 자동 구성을 사용하여 추가 구성 요소 수준 옵션을 구성할 수 있습니다. 이러한 옵션은 Spring-boot 구성 요소 메타데이터 JSON 파일 설명자에 포함되어 있습니다.
- 모든 구성 요소, 데이터 형식 및 언어에 대한 모든 Spring Boot 자동 구성 옵션이 포함된 문서 섹션을 추가했습니다.
- 모든 Camel Spring Boot Starter JAR에는 Spring Boot 자동 구성을 최적화하기 위해 JAR에 **META-INF/spring-autoconfigure-metadata.properties** 파일이 포함되어 있습니다.
- Throttler는 동적 표현식을 기반으로 한 상관관계 그룹을 지원하여 다양한 제한 세트로 메시지를 그룹화할 수 있습니다.
- Hystrix EIP는 이제 Camel의 오류 처리기에 대한 상속을 허용하므로 재전송을 통해 오류 처리를 활성화한 경우 전체 Hystrix EIP 블록을 다시 시도할 수 있습니다.
- SQL 및 ElSql 소비자는 이제 경로 양식에서 동적 쿼리 매개변수를 지원합니다. 이 기능은 간단한 표현식을 사용하여 빈을 호출하도록 제한됩니다.
- swagger-restdsl maven 플러그인은 이제 Swagger 사양 파일에서 DTO 모델 클래스 생성을 지원합니다.
- 다음과 같은 중요한 문제가 해결되었습니다.
  - Aggregator2는 모든 그룹의 강제 완료를 위해 제어 헤더를 전파하지 않도록 수정되었으므로 라우팅 중에 다른 집계기 EIP가 나중에 사용 중인 경우 다시 발생하지 않습니다.
  - 오류 처리기에서 재전송이 실행된 경우 고정 추적기가 작동하지 않습니다.
  - XML 문서에 대한 기본 제공 유형 변환기는 구문 분석 오류를 stdout에 출력할 수 있으며 로깅 API를 사용하여 출력하도록 수정되었습니다.
  - 메시지 본문이 스트리밍 기반인 경우 charset 옵션을 사용하여 파일을 작성하여 파일을 작성하는 것이 작동하지 않습니다.
  - 여러 경로를 통해 라우팅할 때 재사용되지 않도록 Zipkin 루트 ID를 수정하여 단일 상위 범위로 그룹화합니다.
  - HTTP 끝점을 사용할 때 숫자가 있는 IP 주소가 포함된 경우 버그가 있었습니다.
  - 임시 대기열을 통해 요청/거부하고 수동 승인 모드를 사용하는 RabbitMQ의 문제를 해결했습니다. 임시 큐(요청/응답 가능)를 인식하지 못합니다.
  - Allow 헤더 for OPTIONS 요청(예: rest-dsl 사용 시)에서 허용되는 모든 HTTP 동사를 반환하지 않을 수 있는 다양한 HTTP 소비자 구성 요소가 수정되었습니다.
  - FluentProducerTemplate에서 thread- Cryostat 문제를 해결했습니다.

6.2. MAVEN 종속 항목 정보

BOM(Maven bill of materials) 파일의 목적은 잘 작동하는 Maven 종속성 버전 집합을 제공하여 모든 Maven 아티팩트에 대해 버전을 개별적으로 정의할 필요가 없도록 하는 것입니다.

Fuse가 실행되는 각 컨테이너에 전용 BOM 파일이 있습니다.

참고

이러한 BOM 파일을 여기에서 찾을 수 있습니다. https://github.com/jboss-fuse/redhat-fuse. 또는 BOM 파일 업데이트에 대한 정보는 최신 릴리스 노트 로 이동합니다.

Fuse BOM은 다음과 같은 이점을 제공합니다.

- **pom.xml** 파일에 종속성을 추가할 때 버전을 지정할 필요가 없도록 Maven 종속 항목에 대한 버전을 정의합니다.
- 특정 버전의 Fuse에서 완전히 테스트 및 지원되는 선별된 종속성 세트를 정의합니다.
- Fuse 업그레이드 간소화.

중요

Fuse BOM에서 정의한 종속성 세트만 Red Hat에서 지원합니다.

6.3. FUSE 프로젝트의 MAVEN 종속성 업데이트

JBoss EAP용 Fuse 애플리케이션을 업그레이드하려면 프로젝트의 Maven 종속성을 업데이트합니다.

절차

1. 프로젝트의 **pom.xml** 파일을 엽니다.
2. 다음 예와 같이 프로젝트의 **pom.xml** 파일(또는 상위 **pom.xml** 파일)에 **dependencyManagement** 요소를 추가합니다.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<project ...>
  ...
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

    <!-- configure the versions you want to use here -->
    <fuse.version>7.11.1.fuse-sb2-7_11_1-00022-redhat-00002</fuse.version>

  </properties>

  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>org.jboss.redhat-fuse</groupId>
        <artifactId>fuse-eap-bom</artifactId>
        <version>${fuse.version}</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
  ...
</project>
```

3. **pom.xml** 파일을 저장합니다.

pom.xml 파일에서 BOM을 종속성으로 지정하면 아티팩트 버전을 지정하지 *않고* Maven 종속성을 **pom.xml** 파일에 추가할 수 있습니다. 예를 들어 **camel-velocity** 구성 요소에 대한 종속성을 추가하려면 **pom.xml** 파일의 **종속** 항목에 다음 XML 조각을 추가합니다.

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-velocity</artifactId>
  <scope>provided</scope>
</dependency>
```

이 종속성 정의에서 **version** 요소를 생략하는 방법을 참조하십시오.

6.4. JAVA EE 종속 항목 업그레이드

Fuse 7.8에서는 BOM 파일의 일부 관리 종속 항목에 **groupId** 또는 **artifactId** 속성이 업데이트되었으므로 그에 따라 프로젝트의 **pom.xml** 파일을 업데이트해야 합니다.

절차

1. 프로젝트의 **pom.xml** 파일을 엽니다.
2. **org.jboss.spec.javax.** Cryostat 버전을 1.2에서 1.3으로 변경하고 **org.jboss.spec.javax.servlet** 버전을 3.1에서 4.0으로 변경하려면 다음 예와 같이 종속 항목을 업데이트합니다.

```
<dependency>
  <groupId>org.jboss.spec.javax.transaction</groupId>
  <artifactId>jboss-transaction-api_1.3_spec</artifactId>
</dependency>

<dependency>
    <groupId>org.jboss.spec.javax.servlet</groupId>
    <artifactId>jboss-servlet-api_4.0_spec</artifactId>
</dependency>
```

3. Java EE API에서 Jakarta EE로 마이그레이션하려면 **javax.*** 를 각 **groupId** 에 대해 **jakarta.*** 로 바꾸고 다음 예와 같이 개별 종속 항목에 대한 **artifactId** 를 수정합니다.

```
<dependency>
  <groupId>jakarta.validation</groupId>
  <artifactId>jakarta.validation-api</artifactId>
</dependency>

<dependency>
  <groupId>jakarta.enterprise</groupId>
  <artifactId>jakarta.enterprise.cdi-api</artifactId>
</dependency>

<dependency>
  <groupId>jakarta.inject</groupId>
  <artifactId>jakarta.inject-api</artifactId>
</dependency>
```

6.5. JBOSS EAP 설치에서 기존 FUSE 업그레이드

다음 절차에서는 JBoss EAP 설치에서 기존 Fuse를 업그레이드하는 방법을 설명합니다.

절차

1. 하나의 JBoss EAP 마이너 릴리스에서 다른 릴리스로 업그레이드하려면 JBoss EAP 패치 및 업그레이드 가이드 가이드 의 지침을 따라야 합니다.
2. Fuse를 업데이트하려면 JBoss EAP에 설치 가이드에 설명된 대로 Fuse on JBoss EAP 설치 프로그램을 실행해야 합니다.

참고

Fuse 애플리케이션을 다시 컴파일하거나 축소할 필요가 없습니다.

6.6. FUSE 및 JBOSS EAP를 동시에 업그레이드

다음 절차에서는 JBoss EAP 7.2에서 JBoss EAP 7.2의 Fuse 7.7을 JBoss EAP 7.3에서 Fuse 7.8로 마이그레이션하는 경우와 같이 Fuse 설치를 업그레이드하는 방법과 JBoss EAP 런타임 시나리오로 JBoss EAP 런타임을 업그레이드하는 방법을 설명합니다.

주의

Fuse 및 JBoss EAP 런타임을 **모두** 업그레이드할 때 Fuse와 JBoss EAP 런타임을 새로 설치하는 것이 좋습니다.

절차

1. JBoss EAP 런타임을 새로 설치하려면 JBoss EAP 에 설치 가이드의 지침을 따르십시오.
2. Fuse의 새 설치를 수행하려면 JBoss EAP에 설치 가이드에 설명된 대로 Fuse on JBoss EAP 설치 프로그램을 실행합니다.

7장. KARAF 독립 실행형에서 FUSE 애플리케이션 업그레이드

Karaf에서 Fuse 애플리케이션을 업그레이드하려면 다음을 수행합니다.

- 7.1절. "Camel 마이그레이션 고려 사항" 에 설명된 대로 Apache Camel 업데이트를 고려해야 합니다.
- 올바른 Fuse 버전을 사용하고 있는지 확인하려면 Fuse 프로젝트의 Maven 종속성을 업데이트해야 합니다.

일반적으로 Maven을 사용하여 Fuse 애플리케이션을 빌드합니다. Maven은 Apache의 무료 오픈 소스 빌드 툴입니다. Maven 구성은 Fuse 애플리케이션 프로젝트의 **pom.xml** 파일에 정의되어 있습니다. Fuse 프로젝트를 빌드하는 동안 기본 동작은 Maven이 외부 리포지토리를 검색하고 필요한 아티팩트를 다운로드하는 것입니다. Maven 빌드 프로세스에서 올바른 Fuse 지원 아티팩트 세트를 선택할 수 있도록 BOM(Fuse of material)에 대한 종속성을 **pom.xml** 파일에 추가합니다.

다음 섹션에서는 Maven 종속 항목에 대한 정보와 Fuse 프로젝트에서 업데이트하는 방법을 설명합니다.

- 7.2절. "Maven 종속 항목 정보"
- 7.3절. "Fuse 프로젝트의 Maven 종속성 업데이트"

7.1. CAMEL 마이그레이션 고려 사항

MongoClients 팩토리를 사용하여 MongoDB에 연결 생성

Fuse 7.11에서 **com.mongodb.MongoClient** 대신 **com.mongodb.client.MongoClient** 를 사용하여 MongoDB에 대한 연결을 만듭니다(전체 경로에 추가 *.client* 하위 패키지 참조).

기존 Fuse 애플리케이션이 **camel-mongodb** 구성 요소를 사용하는 경우 다음을 수행해야 합니다.

- 애플리케이션을 업데이트하여 연결 빈을 **com.mongodb.client.MongoClient** 인스턴스로 만듭니다.
  예를 들어 다음과 같이 MongoDB에 대한 연결을 생성합니다.

  ```
  import com.mongodb.client.MongoClient;
  ```

  그런 다음 다음 예와 같이 MongoClient 8080을 생성할 수 있습니다.

  ```
  return MongoClients.create("mongodb://admin:password@192.168.99.102:32553");
  ```

- 평가하고 필요한 경우 MongoClient 클래스에서 노출하는 메서드와 관련된 모든 코드를 리팩터링합니다.

Camel 2.23

Red Hat Fuse는 Apache Camel 2.23을 사용합니다. Fuse 7.8로 업그레이드할 때 Camel 2.22 및 2.23에 대한 다음 업데이트를 고려해야 합니다.

Camel 2.22 업데이트

- Camel이 Spring Boot v1에서 v2로 업그레이드되었으므로 v1은 더 이상 지원되지 않습니다.
- Spring Framework 5로 업그레이드 Camel은 Spring 4.3.x와도 함께 작동해야 하지만 향후 Spring 5.x는 향후 릴리스에서 최소 Spring 버전이 될 것입니다.

- Karaf 4.2로 업그레이드했습니다. Camel을 Karaf 4.1에서 실행할 수 있지만 이 릴리스에서 Karaf 4.2만 공식적으로 지원합니다.
- toD DSL을 사용하여 최적화하여 가능한 구성 요소에 대해 끝점 및 생산자를 재사용합니다. 예를 들어 HTTP 기반 구성 요소는 동일한 호스트로 전송되는 동적 URI와 함께 생산자(HTTP 클라이언트)를 재사용합니다.
- 이제 read-lock idempotent/idempotent-changed가 있는 File2 소비자는 이제 파일이 in-process로 간주될 때 릴리스 작업을 지연하도록 구성할 수 있습니다. 이 파일은 프로세스 내/활성 클러스터 설정에서 사용 가능하며, 다른 노드에서 처리된 파일을 처리할 수 있는 파일로 너무 빠르게 표시되지 않도록 합니다(RemoveOnCommit=true).
- 요청/응답 모드에서 Netty4 프로듀서에서 사용자 정의 요청/일반 상관 ID 관리자 구현을 플러그인할 수 있습니다. 이제 Twitter 구성 요소에서 확장 모드를 사용하여 140자를 초과한 event를 지원합니다.
- rest DSL 생산자는 이제 endpointProperties를 사용하여 REST 구성에서 구성할 수 있습니다.
- Kafka 구성 요소에서는 Camel과 Kafka 메시지 간의 헤더 매핑을 제어하기 위한 사용자 지정 구현을 플러그인하도록 HeaderFilterStrategy를 지원합니다.
- REST DSL은 이제 클라이언트 요청 검증을 지원하여 REST 서비스에 Content-Type/Accept 헤더가 가능한지 확인합니다.
- Camel에는 Camel 구현 또는 Spring Cloud를 사용하여 서비스 레지스트리(예: consul, etcd 또는 Zookeeper)에 경로를 등록할 수 있는 Service Registry SPI가 있습니다.
- SEDA 구성 요소의 기본 대기열 크기는 무제한이 아닌 1000입니다.
- 다음과 같은 중요한 문제가 해결되었습니다.
  - CXF 연속 시간제한 문제가 camel-cxf 소비자와 관련된 수정으로 인해 소비자가 시간 초과를 트리거하는 대신 데이터가 포함된 응답을 반환하도록 할 수 있습니다.
  - 고정 camel-cxf 소비자는 강력한 단방향 작업을 사용할 때 UoW를 릴리스하지 않습니다.
  - AdviceWith를 사용하여 수정되었으며 onException에서 weave 메서드를 사용하는 경우 작동하지 않습니다.
  - 병렬 처리 및 스트리밍 모드에서 고정 Splitter는 메시지 본문을 차단하는 동안 처음으로 호출된 next() 메서드 호출에서 예외를 발생시킬 때 차단될 수 있습니다.
  - autoCommitEnable=false인 경우 Kafka 소비자가 자동 커밋되지 않도록 수정했습니다.
  - 고정 파일 소비자는 기본적으로 markerFile을 읽기 잠금으로 사용했으며 이는 none이어야 했습니다.
  - Kafka와의 수동 커밋을 사용하여 이전 레코드 오프셋이 아닌 현재 레코드 오프셋을 제공합니다(및 먼저 -1).
  - Java DSL의 고정 콘텐츠 기반 라우터는 서술자가 있는 경우 속성 자리 표시자를 확인하지 않을 수 있습니다.

Camel 2.23 업데이트

- Spring Boot 2.1으로 업그레이드했습니다.

- 이제 Spring-boot 자동 구성을 사용하여 추가 구성 요소 수준 옵션을 구성할 수 있습니다. 이러한 옵션은 Spring-boot 구성 요소 메타데이터 JSON 파일 설명자에 포함되어 있습니다.
- 모든 구성 요소, 데이터 형식 및 언어에 대한 모든 Spring Boot 자동 구성 옵션이 포함된 문서 섹션을 추가했습니다.
- 모든 Camel Spring Boot Starter JAR에는 Spring Boot 자동 구성을 최적화하기 위해 JAR에 **META-INF/spring-autoconfigure-metadata.properties** 파일이 포함되어 있습니다.
- Throttler는 동적 표현식을 기반으로 한 상관관계 그룹을 지원하여 다양한 제한 세트로 메시지를 그룹화할 수 있습니다.
- Hystrix EIP는 이제 Camel의 오류 처리기에 대한 상속을 허용하므로 재전송을 통해 오류 처리를 활성화한 경우 전체 Hystrix EIP 블록을 다시 시도할 수 있습니다.
- SQL 및 ElSql 소비자는 이제 경로 양식에서 동적 쿼리 매개변수를 지원합니다. 이 기능은 간단한 표현식을 사용하여 빈을 호출하도록 제한됩니다.
- swagger-restdsl maven 플러그인은 이제 Swagger 사양 파일에서 DTO 모델 클래스 생성을 지원합니다.
- 다음과 같은 중요한 문제가 해결되었습니다.
  - Aggregator2는 모든 그룹의 강제 완료를 위해 제어 헤더를 전파하지 않도록 수정되었으므로 라우팅 중에 다른 집계기 EIP가 나중에 사용 중인 경우 다시 발생하지 않습니다.
  - 오류 처리기에서 재전송이 실행된 경우 고정 추적기가 작동하지 않습니다.
  - XML 문서에 대한 기본 제공 유형 변환기는 구문 분석 오류를 stdout에 출력할 수 있으며 로깅 API를 사용하여 출력하도록 수정되었습니다.
  - 메시지 본문이 스트리밍 기반인 경우 charset 옵션을 사용하여 파일을 작성하여 파일을 작성하는 것이 작동하지 않습니다.
  - 여러 경로를 통해 라우팅할 때 재사용되지 않도록 Zipkin 루트 ID를 수정하여 단일 상위 범위로 그룹화합니다.
  - HTTP 끝점을 사용할 때 숫자가 있는 IP 주소가 포함된 경우 버그가 있었습니다.
  - 임시 대기열을 통해 요청/거부하고 수동 승인 모드를 사용하는 RabbitMQ의 문제를 해결했습니다. 임시 큐(요청/응답 가능)를 인식하지 못합니다.
  - Allow 헤더 for OPTIONS 요청(예: rest-dsl 사용 시)에서 허용되는 모든 HTTP 동사를 반환하지 않을 수 있는 다양한 HTTP 소비자 구성 요소가 수정되었습니다.
  - FluentProducerTemplate에서 thread- Cryostat 문제를 해결했습니다.

7.2. MAVEN 종속 항목 정보

BOM(Maven bill of materials) 파일의 목적은 잘 작동하는 Maven 종속성 버전 집합을 제공하여 모든 Maven 아티팩트에 대해 버전을 개별적으로 정의할 필요가 없도록 하는 것입니다.

Fuse가 실행되는 각 컨테이너에 전용 BOM 파일이 있습니다.

참고

이러한 BOM 파일을 여기에서 찾을 수 있습니다. https://github.com/jboss-fuse/redhat-fuse. 또는 BOM 파일 업데이트에 대한 정보는 최신 릴리스 노트 로 이동합니다.

Fuse BOM은 다음과 같은 이점을 제공합니다.

- **pom.xml** 파일에 종속성을 추가할 때 버전을 지정할 필요가 없도록 Maven 종속 항목에 대한 버전을 정의합니다.
- 특정 버전의 Fuse에서 완전히 테스트 및 지원되는 선별된 종속성 세트를 정의합니다.
- Fuse 업그레이드 간소화.

중요

Fuse BOM에서 정의한 종속성 세트만 Red Hat에서 지원합니다.

7.3. FUSE 프로젝트의 MAVEN 종속성 업데이트

Karaf용 Fuse 애플리케이션을 업그레이드하려면 프로젝트의 Maven 종속성을 업데이트합니다.

절차

1. 프로젝트의 **pom.xml** 파일을 엽니다.
2. 다음 예와 같이 프로젝트의 **pom.xml** 파일(또는 상위 **pom.xml** 파일)에 **dependencyManagement** 요소를 추가합니다.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<project ...>
  ...
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

    <!-- configure the versions you want to use here -->
    <fuse.version>7.11.1.fuse-sb2-7_11_1-00022-redhat-00002</fuse.version>

  </properties>

  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>org.jboss.redhat-fuse</groupId>
        <artifactId>fuse-karaf-bom</artifactId>
        <version>${fuse.version}</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
  ...
</project>
```

3. **pom.xml** 파일을 저장합니다.

pom.xml 파일에서 BOM을 종속성으로 지정하면 아티팩트 버전을 지정하지 *않고* Maven 종속성을 **pom.xml** 파일에 추가할 수 있습니다. 예를 들어 **camel-velocity** 구성 요소에 대한 종속성을 추가하려면 **pom.xml** 파일의 **종속** 항목에 다음 XML 조각을 추가합니다.

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-velocity</artifactId>
  <scope>provided</scope>
</dependency>
```

이 종속성 정의에서 **version** 요소를 생략하는 방법을 참조하십시오.

8장. KARAF에서 FUSE STANDALONE 업그레이드

Fuse on Apache Karaf 업그레이드 메커니즘을 사용하면 Karaf에서 업데이트된 버전의 Fuse를 다시 설치할 필요 없이 Apache Karaf 컨테이너에 수정 사항을 적용할 수 있습니다. 또한 업그레이드로 인해 배포된 애플리케이션에 문제가 발생하는 경우 업그레이드를 롤백할 수 있습니다.

업그레이드 설치 프로그램 파일은 Apache Karaf에 Fuse를 설치하는 데 사용하는 *것과 동일한* 파일입니다.

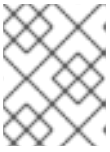
참고

업그레이드 설치 프로그램 파일을 얻으려면 Red Hat 고객 포털의 **다운로드** 페이지로 이동하여 Apache Karaf에서 Fuse용 최신 버전의 설치 아카이브(예: **fuse-karaf-7.11.1.fuse-7_11_1-00013-redhat-00003.zip**)를 다운로드합니다.

8.1. FUSE ON KARAF를 업그레이드할 때의 영향

업그레이드 메커니즘은 **번들 JAR 및 정적 파일(예:etc/ 디렉토리 아래의 구성 파일** 포함)을 포함한 **모든** 설치 파일을 업데이트할 수 있습니다. Apache Karaf 업그레이드 프로세스의 Fuse:

- 번들 JAR, 구성 파일 및 정적 파일을 포함한 모든 파일을 업데이트합니다.
- 현재 컨테이너 인스턴스(및 **데이터/** 디렉터리 아래의 런타임 스토리지)와 기본 설치를 모두 패치합니다. 따라서 컨테이너 인스턴스를 삭제한 후 패치가 유지됩니다.
- 기능 리포지토리 파일 및 기능 자체를 포함하여 Karaf 기능과 관련된 모든 파일을 업데이트합니다. 따라서 롤업 패치 후에 설치된 모든 기능은 올바른 패치된 종속 항목을 참조합니다.
- 필요한 경우 구성 파일(예: **etc/**아래 파일)을 업데이트하고 패치를 통해 변경한 구성 변경 사항을 자동으로 병합합니다. 병합 충돌이 발생하는 경우 패치 로그에서 처리 방법에 대한 자세한 내용을 참조하십시오.
- 대부분의 병합 충돌은 자동으로 해결됩니다. 예를 들어 패치 메커니즘은 속성 파일의 속성 수준에서 충돌을 감지합니다. 사용자인지 또는 속성을 변경한 패치인지 여부를 감지합니다. 한 쪽만 속성을 변경한 경우 변경 사항은 보존됩니다.
- 패치를 롤백할 수 있도록 설치에 수행된 **모든** 변경 사항(정의 파일 포함)을 추적합니다.



참고

롤업 패치 메커니즘은 내부 git 리포지토리(**patches/.management/history**)를 사용하여 변경 사항을 추적합니다.

8.2. KARAF에서 FUSE STANDALONE 업그레이드

다음 지침에서는 Apache Karaf에서 Fuse를 업그레이드하는 방법을 설명합니다. 업그레이드 절차를 시작하기 전에 모든 사전 요구 사항이 완료되었는지 확인합니다.

사전 요구 사항

- 업그레이드하기 전에 Apache Karaf 설치에서 Fuse를 전체 백업해야 합니다.
- 아직 실행되지 않은 경우 컨테이너를 시작합니다.

작은 정보

컨테이너가 백그라운드에서 실행 중인 경우(또는 원격으로) SSH 콘솔 클라이언트인 **bin/client** 를 사용하여 컨테이너에 연결합니다.

- **patch:add** 명령을 호출하여 업그레이드 설치 프로그램 파일을 컨테이너의 환경에 추가합니다. 예를 들어 **fuse-karaf-7.11.1.fuse-7_11_1-00013-redhat-00003.zip** 업그레이드 설치 프로그램 파일을 추가하려면 다음을 수행합니다.

```
patch:add file:///path/to/fuse-karaf-7.11.1.fuse-7_11_1-00013-redhat-00003.zip
```

참고

patch:find 명령은 컨테이너 환경에 최신 핫픽스 패치를 찾아서 추가하는 데만 사용할 수 있습니다. 전체 업그레이드 패치를 적용하는 데 사용할 수 없습니다.

절차

1. **patch:update** 명령을 실행합니다. 컨테이너를 다시 시작할 필요가 없습니다.

```
karaf@root()> patch:update
Current patch mechanism version: 7.1.0.fuse-710023-redhat-00001
New patch mechanism version detected: 7.2.0.fuse-720035-redhat-00001
Uninstalling patch features in version 7.1.0.fuse-710023-redhat-00001
Installing patch features in version 7.2.0.fuse-720035-redhat-00001
```

2. **patch:list** 명령을 호출하여 업그레이드 설치 프로그램 목록을 표시합니다. 이 목록에서 **[name]** 제목 아래의 항목은 업그레이드 ID입니다. 예를 들면 다음과 같습니다.

```
karaf@root()> patch:list
[name]                                  [installed] [rollup] [description]
fuse-karaf-7.2.0.fuse-720035-redhat-00001 false       true     fuse-karaf-7.2.0.fuse-
720035-redhat-00001
```

3. **patch:simulate** 명령을 호출하고 적용할 업그레이드 ID를 다음과 같이 지정하여 업그레이드를 시뮬레이션합니다.

```
karaf@root()> patch:simulate fuse-karaf-7.2.0.fuse-720035-redhat-00001
INFO : org.jboss.fuse.modules.patch.patch-management (226): Installing rollup patch
"fuse-karaf-7.2.0.fuse-720035-redhat-00001"
========== Repositories to remove (9):
 - mvn:io.hawt/hawtio-karaf/2.0.0.fuse-710018-redhat-00002/xml/features
...
========== Repositories to add (9):
 - mvn:io.hawt/hawtio-karaf/2.0.0.fuse-720044-redhat-00001/xml/features
...
========== Repositories to keep (10):
 - mvn:org.apache.activemq/artemis-features/2.4.0.amq-711002-redhat-1/xml/features
...
```

```
========== Features to update (100):
[name]                  [version]                  [new version]
aries-blueprint         4.2.0.fuse-710024-redhat-00002    4.2.0.fuse-720061-redhat-
00001
...
========== Bundles to update as part of features or core bundles (100):
[symbolic name]                                    [version]                  [new location]
io.hawt.hawtio-log                                 2.0.0.fuse-710018-redhat-00002
mvn:io.hawt/hawtio-log/2.0.0.fuse-720044-redhat-00001
...
========== Bundles to reinstall as part of features or core bundles (123):
[symbolic name]                                    [version]                  [location]
com.fasterxml.jackson.core.jackson-annotations     2.8.11
mvn:com.fasterxml.jackson.core/jackson-annotations/2.8.11
...
Simulation only - no files and runtime data will be modified.
karaf@root()>
```

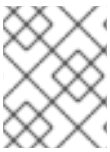
이렇게 하면 업그레이드가 수행되지만 컨테이너를 실제로 변경하지 않는 경우 컨테이너에 대한 변경 로그가 생성됩니다. 시뮬레이션 로그를 검토하여 컨테이너에 적용할 변경 사항을 파악합니다.

4. **patch:install** 명령을 호출하여 컨테이너를 업그레이드하고 적용할 업그레이드 ID를 지정합니다. 예를 들면 다음과 같습니다.

```
karaf@root()> patch:install fuse-karaf-7.11.1.fuse-7_11_1-00013-redhat-00003
```

5. 업그레이드 아티팩트 중 하나를 검색하여 업그레이드를 확인합니다. 예를 들어 Fuse 7.1.0을 Fuse 7.2.0으로 업그레이드한 경우 다음과 같이 빌드 번호인 7_11_1-00017-redhat-00001을 사용하여 번들을 검색할 수 있습니다.

```
karaf@root()> bundle:list -l | grep 7_11_1-00017-redhat-00001
 22 │ Active │  80 │ 7.11.1.fuse-7_11_1-00013-redhat-00003  │
mvn:org.jboss.fuse.modules/fuse-pax-transx-tm-narayana/7.11.1.fuse-7_11_1-00013-
redhat-00003
188 │ Active │  80 │ 7.11.1.fuse-7_11_1-00013-redhat-00003  │
mvn:org.jboss.fuse.modules.patch/patch-commands/7.11.1.fuse-7_11_1-00013-redhat-
00003
```



참고

업그레이드 후 컨테이너를 다시 시작할 때 welcome 배너에 새 버전과 빌드 번호도 표시됩니다.

8.3. KARAF에서 FUSE의 업그레이드 롤백

경우에 따라 업그레이드가 작동하지 않거나 컨테이너에 새로운 문제가 발생할 수 있습니다. 이 경우 **patch:rollback** 명령을 사용하여 업그레이드를 쉽게 롤백하고 시스템을 이전 상태로 복원할 수 있습니다. 이 지침에서는 이 절차를 안내합니다.

사전 요구 사항

- 최근 Karaf에서 Fuse를 업그레이드했습니다.

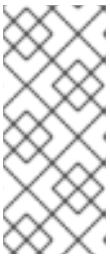
- 업그레이드를 롤백하려고 합니다.

절차

1. **patch:list** 명령을 호출하여 가장 최근에 설치된 패치의 업그레이드 ID **UPGRADE_ID** 를 가져옵니다.

2. 다음과 같이 **patch:rollback** 명령을 호출합니다.

```
patch:rollback UPGRADE_ID
```



참고

경우에 따라 업그레이드를 롤백하기 위해 컨테이너를 다시 시작해야 합니다. 이 경우 컨테이너가 자동으로 다시 시작됩니다. OSGi 런타임의 매우 동적인 특성으로 인해 재시작 중에 호환되지 않는 클래스와 관련된 일부 오류가 표시될 수 있습니다. 이러한 오류는 방금 시작하거나 중지하여 무시해도 되는 OSGi 서비스와 관련이 있습니다.