Red Hat Fuse 7.13

在 Apache Karaf 上安装

在 Apache Karaf 容器上安装红帽 Fuse

Last Updated: 2024-06-12

Red Hat Fuse 7.13 在 Apache Karaf 上安装

在 Apache Karaf 容器上安装红帽 Fuse

法律通告

摘要

易于安装红帽 Fuse，并针对特定环境定制安装。

目录

前言

红帽 Fuse 是一个轻量级、灵活的集成平台，可在扩展企业内部或云中实现快速集成。

基于 Apache Camel，Fuse 利用基于模式的集成、丰富的连接器目录和广泛的数据转换功能，让用户能够集成任何内容。

使开源包含更多

红帽致力于替换我们的代码、文档和 Web 属性中存在问题的语言。我们从这四个术语开始：master、slave、黑名单和白名单。这些更改将在即将发行的几个发行本中逐渐实施。详情请查看我们的 CTO Chris Wright 信息。

第 1 章 在 APACHE KARAF 上安装 FUSE FOR DEVELOPMENT

要开发在 Karaf 上运行的 Fuse 应用程序，请在本地安装 Fuse，如以下主题所述：

- 第 1.1 节 “在 Apache Karaf 上安装 Fuse 的先决条件”
- 第 1.2 节 “在 Apache Karaf 上安装 Fuse”
- 第 1.3 节 “关于在 Karaf 离线上运行 Fuse”
- 第 1.4 节 “（可选）使用独立 Apache 发行版本”

1.1. 在 APACHE KARAF 上安装 FUSE 的先决条件

先决条件

在 Apache Karaf 上安装 Fuse 前，请确定您的系统：

1. 满足硬件要求。
2. 是一款受支持的平台。
3. 具有受支持的 Java 运行时。
4. 运行支持的标准软件。

1.1.1. Apache Karaf 硬件要求上的 Fuse

在 Apache Karaf 上完全安装 Fuse 的硬件要求包括：

- 250 MB 可用磁盘空间
- 2 GB RAM

此外，运行 Fuse 的系统还需要空间进行缓存、持久消息存储和其他功能。实际要求取决于您的 Fuse 应用程序的作用。

1.1.2. Apache Karaf 支持的 Fuse

1. 确认您要在其上安装 Fuse 的系统正在运行受支持的平台。红帽在 Red Hat Fuse 支持的配置 中列出的平台上测试并支持 Fuse 产品。

1.1.3. Apache Karaf 上的 Fuse 支持 Java 运行时

有关支持的运行时版本列表，请参阅 Red Hat Fuse 支持的配置 中支持的 Java 版本部分。

注意

您不能在包含空格的目录路径中安装 Java 运行时。路径中的空格会导致在运行时在 Apache Karaf 上的 Fuse 中无法预计的错误。

例如，**C:\Program Files\Java\jdk8** 不是可接受的路径。

1.1.4. Apache Karaf 上的 Fuse 支持标准

有关支持的标准软件列表，请参阅 Red Hat Fuse 支持的标准 ：

- Web 服务
- API
- 传输协议.

1.2. 在 APACHE KARAF 上安装 FUSE

红帽客户门户网站上 Fuse 7.13 的标准安装软件包可从红帽客户门户下载。它安装 Apache Karaf 容器的标准装配，并提供完整的 Fuse 技术堆栈。

可以创建自己的自定义装配 Fuse 7.13，其中包含 Fuse 功能和捆绑包的自定义子集。**自定义** 快速入门演示了如何使用 Maven 创建 Red Hat Fuse 的自定义装配。您可以从 Fuse Software Downloads 页面提供的可下载文件安装所有快速入门。

前提条件

安装 Fuse 的系统必须满足 第 1.1 节 “在 Apache Karaf 上安装 Fuse 的先决条件” 中描述的硬件和软件要求。

流程

1. 在浏览器中，进入 Fuse Software Downloads 页面。
   如果您还没有登录到红帽客户门户网站，则会显示一个提示登录，然后显示下载页面（您的帐户必须与红帽 Fuse 订阅关联）。
2. 在 Fuse **Software Downloads** 页面中，在 **Karaf Installer 上的 Red Hat Fuse 7.13**右侧，单击 **Download** 以获取本地 zip 文件。
3. 将 zip 文件的内容提取到您拥有所有权限的目录中。
   不要将 zip 文件解压缩到在路径名中包含空格或任何以下特殊字符的目录中：{ , **% , ^**, **"**.例如，不要解包到 **C:\Documents and Settings\Greco#Roman\Desktop\fuse**。
4. 如果使用 IBM JDK：

   例 1.1. IBM JDK 的额外步骤

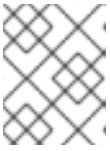
   a. 在 Fuse 安装目录中，在 **/lib/endorsed** 目录中，删除 **saaj-api .jar** 文件，例如：

   ```
   rm lib/endorsed/org.apache.servicemix.specs.saaj-api-1.3-2.9.0.jar
   ```

   b. 设置 **JAVA_OPTS** 环境变量，如下所示：

   ```
   JAVA_OPTS=-Xshareclasses:none
   ```

   在启动 Karaf 容器前，您必须设置 **JAVA_OPTS** 环境变量。
5. 添加管理员用户，以启用对 Karaf 容器上的 Fuse 的远程访问，并访问 Fuse 控制台。



注意

默认情况下，没有为容器定义任何用户。在这种情况下，您可以在前台运行容器，但您无法远程访问容器，您无法在后台运行容器。

按照以下步骤，至少创建一个具有 **admin** 角色的用户：

a. 在文本编辑器中，打开 **etc/users.properties** 文件，该文件位于 Fuse 安装目录中。

b. 找到以下行：

```
#admin = admin,_g_:admingroup
#_g_\:admingroup = group,admin,manager,viewer,systembundles,ssh
```

c. 对于每行，删除前导 **#** 字符以取消注释行。

d. 在第一行中，将 **admin** 的第一个实例更改为您选择的用户名，如 **user1**。

e. 在同一行中，将 **admin** 的第二个实例更改为该用户的密码，如 **passw0rd**。
例如：

```
user1 = passw0rd,_g_:admingroup
_g_\:admingroup = group,admin,manager,viewer,systembundles,ssh
```

f. 保存并关闭该文件。

6. 要启动 Fuse，请在 Windows 上运行 **bin/fuse** on Linux/Unix 或 **bin\fuse.bat**。

7. 另外，要访问 Fuse 控制台，请在网页浏览器中打开提供的 URL，并使用您在 **etc/users.properties** 文件中设置的用户名和密码登录。有关使用 Fuse 控制台的更多信息，请参阅在 Karaf Standalone 上管理 Fuse。

1.3. 关于在 KARAF 离线上运行 FUSE

您可以以离线模式运行 Apache Karaf 容器，即没有互联网连接。但是，如果您计划将自定义应用程序部署到容器中，可能需要将这些依赖项下载到本地 Maven 存储库，然后才能使用这些应用程序在离线模式下运行容器。

要以离线模式运行 Apache Karaf 容器，需要区分以下类型的依赖项：

- **运行时依赖项** 是在默认配置中运行 Apache Karaf 容器所需的依赖项。
- 构建时依赖项是构建自定义应用所需的依赖项，其中可能包括第三方库。

以下是在离线模式下可以进行的操作摘要，以及需要在在线模式下完成的内容（通过互联网连接）：

- **在默认配置中运行 Apache Karaf 容器，支持离线模式。Apache Karaf 容器的默认配置由 etc/org.apache.karaf.features.cfg 文件中的 featuresBoot 属性指定。所需的依赖项在安装的 system/ 子目录中提供。**
- **通常，在离线模式 中不支持 安装附加功能。在原则上，您可以使用 features:install 命令从标准功能存储库安装任何功能（如由 etc/org.apache.karaf.features.cfg 文件中的 featuresRepositories 属性指定），因此不支持离线模式。**
- **通常，在离线模式 中不支持 部署自定义应用程序。有些情况下，带有最小构建时依赖项的应用程序会离线部署。但是，自定义应用通常具有需要互联网连接的第三方依赖项，以便 Apache Maven 可以下载 JAR 文件。**

其他资源

第 3.3 节 “使用本地 Maven 存储库”

1.4. （可选）使用独立 APACHE 发行版本

Red Hat Fuse 提供下载的额外软件包，其中包含 Apache Camel 和 Apache CXF 标准发行版。如果要使用 Apache Camel 或 Apache CXF 的标准上游分发，请使用下载的 extras 软件包中的存档版本。

流程

1. 登录红帽客户门户。
2. 访问 红帽客户门户网站→Downloads→Red Hat Fuse→Downloads 页。
3. 从 Software Downloads 页面上的 Version 下拉列表中选择 7.13.0。
4. 下载 Red Hat Fuse 7.13.0 Extras 存档。
   extras 归档文件包含以下存档文件嵌套在其中：
   - apache-camel-2.23.2.fuse-7_13_0-00013-redhat-00001.zip
   - apache-cxf-3.3.6.fuse-7_13_0-00015-redhat-00001.zip
5. 将这些文件复制到所需位置，并使用适合您的平台的工具解压缩它们。

警告

不要将存档文件解压缩到其路径名称中有空格的文件夹。例如，请勿解压缩到 C:\Documents and Settings\Greco Roman\Desktop\fuse。

第 2 章 在 APACHE KARAF 上将 HOTFIX PATCH 应用到 FUSE

2.1. 修补功能和捆绑包

补丁是 ZIP 存档，其中包含 Apache Karaf 安装中 Fuse 中更新的文件版本。它们是：

- 捆绑包：它们是最常见的，在最简单的情形中，热修补可以包括单一捆绑包。
- 分别存在于 $FUSE_HOME/etc 和 $FUSE_HOME/bin 目录中的配置文件和脚本。
- 不是普通捆绑包的库，并位于 $FUSE_HOME/lib 目录中。
- 功能定义更改：通常的 Karaf 功能包含在 $FUSE_HOME/system 目录中可用的描述符中，但热修补程序不会更改这些文件。相反，热修复补丁可能会更改功能覆盖文件，即 $FUSE_HOME/etc/org.apache.karaf.features.xml。这可让您通过升级给定的功能的捆绑包，甚至使用其他捆绑包来以热修补代码方式更改功能定义。

升级和热修复补丁之间的区别

- 修补程序补丁：修补程序仅包含针对一个或多个关键漏洞的修复。它们适用于您当前的 Red Hat Fuse 发行版。它的主要目的是更新现有发行版中的一些捆绑包和库。
- 升级：Apache Karaf 升级机制上的 Fuse 可让您对 Apache Karaf 容器应用修复，而无需在 Karaf 上重新安装 Fuse 的更新版本。如果升级导致了与部署的应用程序相关的问题，它还允许您回滚升级。Apache Karaf 升级过程上的 Fuse 会更新任何文件，包括捆绑 JAR、配置文件和任何静态文件。

对于 Apache Karaf Standalone 上的 Fuse，您可以使用 Karaf 控制台补丁 shell 中的命令应用补丁。这个方法不可破坏性且不可逆。以下流程也可用于升级 Apache Karaf 上的红帽 Fuse。有关升级的更多信息，请参阅 在 Apache Karaf 上升级 Fuse。

2.2. 在 APACHE KARAF 上将 HOTFIX PATCH 应用到 RED HAT FUSE

您可以使用热修补机制同时更新可用的功能定义和捆绑包。将热修复补丁应用到 Apache Karaf 安装上的 Fuse 的步骤如下：

流程

1. 在升级前，在 Apache Karaf 安装上对 Fuse 进行完整备份。
2. 打开一个终端，并在 Apache karaf 服务器上启动 Fuse。

```
[user@FUSE_HOME/bin ~] $ ./fuse
```

3. 可选：从客户门户网站下载所需的补丁并跳至第 5 步。
4. 输入 patch:find 命令在 Maven 存储库中找到可用补丁。例如：

```
karaf@root()> patch:find
Found new remote patch at mvn:org.jboss.redhat-fuse/fuse-karaf-patch-repository/7.8.0.fuse-sb2-780040/zip
You can add the patch using "patch:add mvn:org.jboss.redhat-fuse/fuse-karaf-patch-repository/7.8.0.fuse-sb2-780040/zip" command, or simply use "patch:find --add" option.
```

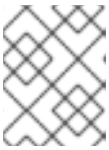
注意

您可以使用 patch:find 命令和 --add 选项来查找最新的补丁并将其添加到容器环境中。

5. **输入 patch:add 命令，将补丁添加到容器环境中。例如：**

```
karaf@root()> patch:add mvn:org.jboss.redhat-fuse/fuse-karaf-patch-repository/7.8.0.fuse-sb2-780040/zip
[name]                                          [installed] [rollup] [description]
[CVEs]
fuse-karaf-maintenance-patch-7.8.0.fuse-sb2-780040 false       false    fuse-karaf-maintenance-patch-7.8.0.fuse-sb2-780040 CVE-2020-28052

Current patch mechanism version: 7.8.0.fuse-780038
New patch mechanism version detected: 7.8.0.fuse-780040
Please run "patch:update" command to upgrade patching mechanism to version 7.8.0.fuse780040
```



注意

除了使用 patch:add 命令外，您还可以通过将 .zip 补丁文件复制到 FUSE_HOME/patches 目录中来自动添加补丁文件。

6. **可选：输入 patch:update 命令来更新补丁机制本身：**

```
karaf@root()> patch:update
Current patch mechanism version: 7.8.0.fuse-780038
New patch mechanism version detected: 7.8.0.fuse-780040
Uninstalling patch features in version 7.8.0.fuse-780038
Installing patch features in version 7.8.0.fuse-780040
```

7. **输入 patch:simulate 命令模拟安装补丁。**
这会生成安装补丁时将对容器进行的更改日志，但不会对容器进行任何更改。检查模拟日志以了解这些更改。

8. **输入 patch:list 命令来查看添加的补丁列表。在此列表中，[name] 标题下的条目是补丁 ID。**

```
karaf@root()> patch:list
[name]                                          [installed] [rollup] [description]
[CVEs]
fuse-karaf-maintenance-patch-7.8.0.fuse-sb2-780040 false       false    fuse-karaf-maintenance-patch-7.8.0.fuse-sb2-780040 CVE-2020-28052
```

9. **如果补丁包含明确的 CVE 元数据，您可以输入 patch:show 命令来查看更多详细信息：**

```
karaf@root()> patch:show fuse-karaf-maintenance-patch-7.8.0.fuse-sb2-780040
Patch ID: fuse-karaf-maintenance-patch-7.8.0.fuse-sb2-780040
Patch Commit ID: a2d7cf58e21116cde66c97232aea4be1ec304400
#### 1 CVE fix:
 - CVE-2020-28052: bouncycastle: password bypass in OpenBSDBCrypt.checkPassword utility possible
   Bugzilla link: https://bugzilla.redhat.com/show_bug.cgi?id=1912881
   CVE link: https://cve.mitre.org/cgi-bin/cvename.cgi?name=2020-28052
```

10. **输入 patch:install 命令并为您要应用的补丁指定补丁 ID，将补丁应用到容器。例如：**

```
patch:install fuse-karaf-maintenance-patch-7.8.0.fuse-sb2-780040
```

2.3. 回滚补丁

您可以使用 patch:rollback 命令回滚已安装的热修复补丁并将其恢复到预补丁行为，如下所示：

流程

1. **输入 patch:list 命令以获取最近安装的补丁的补丁 ID。**
2. **要回滚更新的捆绑包，请输入以下命令：**

```
karaf@root()> patch:rollback my-patch-x
INFO : org.jboss.fuse.modules.patch.patch-management (2): Rolling back non-rollup patch
"my-patch-x"
removing overriden feature: hawtio-rbac/2.0.0.fuse-000117
refreshing features
Enter feature:info command to view the information about the feature.
```

```
karaf@root()> feature:info hawtio-rbac
Feature hawtio-rbac 2.0.0.fuse-000117
Details:
    Installs the hawtio RBAC enabler bundle(s)
Feature has no configuration
Feature has no configuration files
Feature has no dependencies.
Feature contains followed bundles:
    mvn:io.hawt/hawtio-osgi-jmx/2.0.0.fuse-000117
Feature has no conditionals.
```

2.4. 修补 RED HAT FUSE 应用程序

使用新的 patch-maven-plugin 机制，您可以对 Red Hat Fuse 应用程序应用补丁。此机制允许您更改由不同 Red Hat Fuse BOMS 提供的独立版本，如 fuse-springboot-bom 和 fuse-karaf-bom。

2.4.1. 关于 patch-maven-plugin

patch-maven-plugin 执行以下操作：

- **检索与当前 Red Hat Fuse BOM 相关的补丁元数据。**
- **将版本更改应用到从 BOMs 导入的 <dependencyManagement >。**

在 patch-maven-plugin 获取元数据后，它会迭代声明插件的项目的所有受管和直接依赖项，并使用 CVE/patch 元数据替换依赖项版本（如果匹配）。替换了版本后，Maven 构建将继续，并通过标准 Maven 项目阶段进行进度。

2.4.2. 将补丁应用到 Red Hat Fuse 应用程序

patch-maven-plugin 的目的是将 Red Hat Fuse BOM 中列出的依赖项版本更新为您要应用到应用程序的补丁元数据中指定的版本。

流程

以下流程解释了如何将补丁应用到您的应用程序。

1. 将 patch-maven-plugin 添加到项目的 pom.xml 文件中。patch-maven-plugin 的版本必须与 Fuse BOM 的版本相同。

```
<build>
  <plugins>
    <plugin>
      <groupId>org.jboss.redhat-fuse</groupId>
      <artifactId>patch-maven-plugin</artifactId>
      <version>${version.org.jboss-redhat-fuse}</version>
      <extensions>true</extensions>
    </plugin>
  </plugins>
</build>
```

2. 当您运行 mvn clean deploy 或 mvn dependency:tree 命令中的任何一个时，插件会搜索项目模块来检查是否使用其中一个 Red Hat Fuse BOM。只有两个被认为是支持的 BOM：

   - org.jboss.redhat-fuse:fuse-karaf-bom: 用于 Fuse Karaf BOM
   - org.jboss.redhat-fuse:fuse-springboot-bom: 用于 Fuse Spring Boot BOM

3. 如果没有找到以上 BOM，插件将显示以下信息：

```
$ mvn clean install
[INFO] Scanning for projects...
[INFO]

========== Red Hat Fuse Maven patching ==========

[INFO] [PATCH] No project in the reactor uses Fuse Karaf or Fuse Spring Boot BOM.
Skipping patch processing.
[INFO] [PATCH] Done in 3ms
```

4. 如果同时找到了两个 Fuse BOM，patch-maven-plugin 会停止，并显示以下警告：

```
$ mvn clean install
[INFO] Scanning for projects...
[INFO]

========== Red Hat Fuse Maven patching ==========

[WARNING] [PATCH] Reactor uses both Fuse Karaf and Fuse Spring Boot BOMs. Please
use only one. Skipping patch processing.
[INFO] [PATCH] Done in 3ms
```

5. patch-maven-plugin 尝试获取以下 Maven 元数据值之一。

- 对于使用 Fuse Karaf BOM 的项目，**org.jboss.redhat-fuse/fuse-karaf-patch-metadata/maven-metadata.xml 已解决。这是带有 org.jboss.redhat-fuse:fuse-karaf-patch-metadata:RELEASE 的工件的元数据。**

- 对于使用 Fuse Spring Boot BOM 项目的项目，**org.jboss.redhat-fuse/fuse-springboot-patch-metadata/maven-metadata.xml 已解决。这是带有 org.jboss.redhat-fuse:fuse-springboot-patch-metadata:RELEASE 的工件的元数据。**

  Maven 生成的元数据示例

```
<?xml version="1.0" encoding="UTF-8"?>
<metadata>
  <groupId>org.jboss.redhat-fuse</groupId>
  <artifactId>fuse-springboot-patch-metadata</artifactId>
  <versioning>
    <release>7.8.1.fuse-sb2-781025</release>
    <versions>
      <version>7.8.0.fuse-sb2-780025</version>
      <version>7.7.0.fuse-sb2-770010</version>
      <version>7.7.0.fuse-770010</version>
      <version>7.8.1.fuse-sb2-781025</version>
    </versions>
    <lastUpdated>20201023131724</lastUpdated>
  </versioning>
</metadata>
```

6. **patch-maven-plugin 解析元数据，以选择适用于当前项目的版本。这只适用于使用带有版本 7.8.xxx 的 Fuse BOM 的 Maven 项目。只有与版本范围 7.8、7.9 或更高版本匹配的元数据才适用，且只获取最新版本的元数据。**

7. **patch-maven-plugin 收集在下载由 groupId、artifactId 和版本中找到的groupId、artifactId 和版本 标识时使用的远程 Maven 存储库列表。这些 Maven 存储库是活跃配置集的项目 < repositories> 元素中列出的 Maven 存储库，以及来自settings.xml 文件中的存储库。**

```
$ mvn clean install
[INFO] Scanning for projects...
[INFO]

========== Red Hat Fuse Maven patching ==========

[INFO] [PATCH] Reading patch metadata and artifacts from 2 project repositories
[INFO] [PATCH]  - local-nexus: http://everfree.forest:8081/repository/maven-releases/
[INFO] [PATCH]  - central: https://repo.maven.apache.org/maven2
Downloading from local-nexus: http://everfree.forest:8081/repository/maven-releases/org/jboss/redhat-fuse/fuse-springboot-patch-metadata/maven-metadata.xml
...
```

8. **另外，如果要使用离线存储库，您可以使用 -Dpatch 选项指定由 fuse-karaf/fuse-karaf-patch-repository 或 fuse-springboot/fuse-springboot-patch - repository 模块生成的 ZIP 文件。这些 ZIP 文件与 Maven 存储库结构具有相同的内部结构。例如，**

```
$ mvn clean install -Dpatch=../../../test/resources/patch-3.zip
[INFO] Scanning for projects...
[INFO]
```

```
========== Red Hat Fuse Maven patching ==========

[INFO] [PATCH] Reading metadata and artifacts from /data/sources/github.com/jboss-
fuse/redhat-fuse/fuse-tools/patch-maven-plugin/src/test/resources/patch-3.zip
Downloading from fuse-patch: zip:file:/tmp/patch-3.zip-
1742974214598205745/org/jboss/redhat-fuse/fuse-springboot-patch-metadata/maven-
metadata.xml
Downloaded from fuse-patch: zip:file:/tmp/patch-3.zip-
1742974214598205745/org/jboss/redhat-fuse/fuse-springboot-patch-metadata/maven-
metadata.xml (406 B at 16 kB/s)
Downloading from fuse-patch: zip:file:/tmp/patch-3.zip-
1742974214598205745/org/jboss/redhat-fuse/fuse-springboot-patch-metadata/7.8.0.fuse-
sb2-781023/fuse-springboot-patch-metadata-7.8.0.fuse-sb2-781023.xml
Downloaded from fuse-patch: zip:file:/tmp/patch-3.zip-
1742974214598205745/org/jboss/redhat-fuse/fuse-springboot-patch-metadata/7.8.0.fuse-
sb2-781023/fuse-springboot-patch-metadata-7.8.0.fuse-sb2-781023.xml (926 B at 309 kB/s)
[INFO] [PATCH] Resolved patch descriptor: /home/user/.m2/repository/org/jboss/redhat-
fuse/fuse-springboot-patch-metadata/7.8.0.fuse-sb2-781023/fuse-springboot-patch-
metadata-7.8.0.fuse-sb2-781023.xml
...
```

9. **元数据是否来自远程存储库、本地存储库还是 ZIP 文件，它由 patch-maven-plugin 进行分析。获取的元数据包含 CVE 列表，以及每个 CVE 都有一个受影响的 Maven 工件列表（由 glob 模式和版本范围指定），以及一个包含给定 CVE 修复的版本。例如，**

```
<?xml version="1.0" encoding="UTF-8" ?>

<metadata xmlns="urn:redhat:fuse:patch-metadata:1">
   <product-bom groupId="org.jboss.redhat-fuse" artifactId="fuse-springboot-bom" versions="
[7.8,7.9)" />
   <cves>
      <cve id="CVE-2020-xyz" description="Jetty can be configured to listen on port 8080"
           cve-link="https://nvd.nist.gov/vuln/detail/CVE-2020-xyz"
           bz-link="https://bugzilla.redhat.com/show_bug.cgi?id=42">
         <affects groupId="org.eclipse.jetty" artifactId="jetty-*" versions="[9.4,9.4.32)"
fix="9.4.32.v20200930" />
         <affects groupId="org.eclipse.jetty.http2" artifactId="http2-*" versions="[9.4,9.4.32)"
fix="9.4.32.v20200930" />
      </cve>
   </cves>
   <fixes />
</metadata>
```

10. **最后，当迭代当前项目中的所有受管依赖项时，会查阅补丁元数据中指定的修复列表。这些匹配的依赖项（和受管依赖项）会改为固定版本。例如：**

```
$ mvn clean install -U
[INFO] Scanning for projects...
[INFO

========== Red Hat Fuse Maven patching ==========

[INFO] [PATCH] Reading patch metadata and artifacts from 2 project repositories
[INFO] [PATCH]  - local-nexus: http://everfree.forest:8081/repository/maven-releases/
[INFO] [PATCH]  - central: https://repo.maven.apache.org/maven2
```

```
Downloading from local-nexus: http://everfree.forest:8081/repository/maven-
releases/org/jboss/redhat-fuse/fuse-springboot-patch-metadata/maven-metadata.xml
Downloading from central: https://repo.maven.apache.org/maven2/org/jboss/redhat-
fuse/fuse-springboot-patch-metadata/maven-metadata.xml
Downloaded from local-nexus: http://everfree.forest:8081/repository/maven-
releases/org/jboss/redhat-fuse/fuse-springboot-patch-metadata/maven-metadata.xml (363 B
at 4.3 kB/s)
[INFO] [PATCH] Resolved patch descriptor: /home/user/.m2/repository/org/jboss/redhat-
fuse/fuse-springboot-patch-metadata/7.8.0.fuse-sb2-780032/fuse-springboot-patch-
metadata-7.8.0.fuse-sb2-780032.xml
[INFO] [PATCH] Patch metadata found for org.jboss.redhat-fuse/fuse-springboot-
bom/[7.8,7.9)
[INFO] [PATCH]  - patch contains 1 CVE fix
[INFO] [PATCH] Processing managed dependencies to apply CVE fixes...
(https://nvd.nist.gov/vuln/detail/CVE-2020-xyz, https://bugzilla.redhat.com/show_bug.cgi?
id=42_
[INFO] [PATCH] - CVE-2020-xyz: Jetty can be configured to expose itself on port 8080
[INFO] [PATCH]   Applying change org.eclipse.jetty/jetty-*/[9.4,9.4.32) -> 9.4.32.v20200930
[INFO] [PATCH]    - managed dependency: org.eclipse.jetty/jetty-alpn-
client/9.4.30.v20200611 -> 9.4.32.v20200930
...
[INFO] [PATCH]    - managed dependency: org.eclipse.jetty/jetty-openid/9.4.30.v20200611 ->
9.4.32.v20200930
[INFO] [PATCH]   Applying change org.eclipse.jetty.http2/http2-*/[9.4,9.4.32) ->
9.4.32.v20200930
[INFO] [PATCH]    - managed dependency: org.eclipse.jetty.http2/http2-
client/9.4.30.v20200611 -> 9.4.32.v20200930
...
[INFO] [PATCH] Done in 635ms

================================================
```

跳过补丁

如果您不希望将特定补丁应用到项目，则 patch-maven-plugin 会提供 skip 选项。假设已将 patch-maven-plugin 添加到项目的 pom.xml 文件中，并且您不想更改版本，您可以使用以下任一方法跳过补丁。

- **将 skip 选项添加到项目的 pom.xml 文件中，如下所示：**

```
<build>
  <plugins>
    <plugin>
      <groupId>org.jboss.redhat-fuse</groupId>
      <artifactId>patch-maven-plugin</artifactId>
      <version>${version.org.jboss-redhat-fuse}</version>
      <extensions>true</extensions>
      <configuration>
        <skip>true</skip>
      </configuration>
    </plugin>
  </plugins>
</build>
```

- **运行 mvn 命令时或使用 -DskipPatch 选项，如下所示：**

```
$ mvn dependency:tree -DskipPatch
[INFO] Scanning for projects...
[INFO]
[INFO] ------< org.jboss.redhat-fuse:cve-dependency-management-module1 >-------
[INFO] Building cve-dependency-management-module1 7.8.0.fuse-sb2-780033
[INFO] --------------------------------[ jar ]---------------------------------
...
```

如上述输出中所示，patch-maven-plugin 没有调用，这会导致补丁不会应用到应用。

第 3 章 在本地设置 MAVEN

典型的 Fuse 应用程序开发使用 Maven 来构建和管理项目。

以下主题描述了如何在本地设置 Maven：

- 第 3.1 节 “准备设置 Maven”
- 第 3.2 节 “将红帽软件仓库添加到 Maven”
- 第 3.3 节 “使用本地 Maven 存储库”
- 第 3.4 节 “使用环境变量或系统属性设置 Maven 镜像”
- 第 3.5 节 “关于 Maven 工件和协调”

3.1. 准备设置 MAVEN

Maven 是一个来自 Apache 的免费开源构建工具。通常，您可以使用 Maven 构建 Fuse 应用程序。

流程

1. 从 Maven 下载页面 下载最新版本的 Maven。
2. 确定您的系统已连接到互联网。
   在构建项目时，默认行为是 Maven 搜索外部存储库并下载所需的工件。Maven 查找可通过互联网访问的存储库。

   您可以更改此行为，以便 Maven 仅搜索位于本地网络上的存储库。也就是说，Maven 可以在离线模式下运行。在离线模式中，Maven 会在其本地存储库中查找工件。请参阅 第 3.3 节 “使用本地 Maven 存储库”。

3.2. 将红帽软件仓库添加到 MAVEN

要访问位于 Red Hat Maven 存储库中的工件，您需要将这些存储库添加到 Maven 的 settings.xml 文件中。Maven 在用户主目录的 .m2 目录中查找 settings.xml 文件。如果没有用户指定的 settings.xml 文件，Maven 将使用 M2_HOME/conf/ settings.xml 中的系统级 settings.xml 文件。

前提条件

您知道要在其中添加红帽软件仓库的 settings.xml 文件的位置。

流程

在 settings.xml 文件中，为红帽 软件仓库 添加软件仓库元素，如下例所示：

```
<?xml version="1.0"?>
<settings>

  <profiles>
    <profile>
      <id>extra-repos</id>
      <activation>
        <activeByDefault>true</activeByDefault>
      </activation>
```

```
    <repositories>
     <repository>
         <id>redhat-ga-repository</id>
         <url>https://maven.repository.redhat.com/ga</url>
         <releases>
            <enabled>true</enabled>
         </releases>
         <snapshots>
            <enabled>false</enabled>
         </snapshots>
      </repository>
      <repository>
         <id>redhat-ea-repository</id>
         <url>https://maven.repository.redhat.com/earlyaccess/all</url>
         <releases>
            <enabled>true</enabled>
         </releases>
         <snapshots>
            <enabled>false</enabled>
         </snapshots>
      </repository>
      <repository>
        <id>jboss-public</id>
        <name>JBoss Public Repository Group</name>
        <url>https://repository.jboss.org/nexus/content/groups/public/</url>
      </repository>
    </repositories>
    <pluginRepositories>
      <pluginRepository>
         <id>redhat-ga-repository</id>
         <url>https://maven.repository.redhat.com/ga</url>
         <releases>
            <enabled>true</enabled>
         </releases>
         <snapshots>
            <enabled>false</enabled>
         </snapshots>
      </pluginRepository>
      <pluginRepository>
         <id>redhat-ea-repository</id>
         <url>https://maven.repository.redhat.com/earlyaccess/all</url>
         <releases>
            <enabled>true</enabled>
         </releases>
         <snapshots>
            <enabled>false</enabled>
         </snapshots>
      </pluginRepository>
      <pluginRepository>
        <id>jboss-public</id>
        <name>JBoss Public Repository Group</name>
        <url>https://repository.jboss.org/nexus/content/groups/public</url>
      </pluginRepository>
    </pluginRepositories>
  </profile>
</profiles>
```

```
  <activeProfiles>
    <activeProfile>extra-repos</activeProfile>
  </activeProfiles>

</settings>
```

3.3. 使用本地 MAVEN 存储库

如果您在没有互联网连接的情况下运行容器，并且需要部署一个具有离线依赖项的应用程序，您可以使用 Maven 依赖项插件将应用的依赖项下载到 Maven 离线存储库中。然后，您可以将此自定义 Maven 离线存储库分发到没有互联网连接的机器。

流程

1. **在包含 pom.xml 文件的项目目录中，运行以下命令来下载 Maven 项目的存储库，如下所示：**

   ```
   mvn org.apache.maven.plugins:maven-dependency-plugin:3.1.0:go-offline -
   Dmaven.repo.local=/tmp/my-project
   ```

   在本例中，构建项目所需的 Maven 依赖项和插件将下载到 /tmp/my-project 目录中。

2. **将此自定义 Maven 离线存储库在内部分发到没有互联网连接的任何机器。**

3.4. 使用环境变量或系统属性设置 MAVEN 镜像

在运行应用程序时，您需要访问 Red Hat Maven 存储库中的工件。这些存储库添加到 Maven 的 settings.xml 文件中。Maven 检查settings.xml 文件的以下位置：

- **查找指定的 url**
- **if not found looks for ${user.home}/.m2/settings.xml**
- **如果没有找到 ${maven.home}/conf/settings.xml的查找**
- **如果未找到，则查找 ${M2_HOME}/conf/settings.xml**
- **如果没有找到位置，则会创建空的 org.apache.maven.settings.Settings 实例。**

3.4.1. 关于 Maven 镜像

Maven 使用一组远程存储库来访问工件，这些工件目前在本地存储库中不可用。存储库列表几乎始终包含 Maven Central 存储库，但对于 Red Hat Fuse，它还包含 Maven 红帽存储库。在某些情况下，无法访问不同的远程存储库，您可以使用 Maven 镜像的机制。镜像将特定的存储库 URL 替换为不同的存储库 URL，因此搜索远程工件时的所有 HTTP 流量都可以定向到单个 URL。

3.4.2. 在 settings.xml中添加 Maven 镜像

要设置 Maven 镜像，请在 Maven 的 settings.xml 中添加以下部分：

```
<mirror>
    <id>all</id>
    <mirrorOf>*</mirrorOf>
```

```
    <url>http://host:port/path</url>
</mirror>
```

如果在 settings.xml 文件中找不到以上部分，则不会使用 mirror。要在不提供 XML 配置的情况下指定全局镜像，您可以使用系统属性或环境变量。

3.4.3. 使用环境变量或系统属性设置 Maven 镜像

要使用环境变量或系统属性设置 Maven 镜像，您可以添加：

- **名为 MAVEN_MIRROR_URL 的环境变量到 bin/setenv 文件**
- **名为 mavenMirrorUrl to etc/system.properties 文件的系统属性**

3.4.4. 使用 Maven 选项指定 Maven 镜像 url

要使用替代的 Maven 镜像 url，在环境变量或系统属性之外，在运行应用程序时使用以下 maven 选项：

- **-DmavenMirrorUrl=mirrorId::mirrorUrl**
  for example, **-DmavenMirrorUrl=my-mirror::http://mirror.net/repository**
- **-DmavenMirrorUrl=mirrorUrl**
  例如，-DmavenMirrorUrl=http://mirror.net/repository。在本例中，<mirror> **的** <id> **只是一个镜像(mirror)。**

3.5. 关于 MAVEN 工件和协调

在 Maven 构建系统中，基本构建块是一个 *工件*。构建后，工件的输出通常是一个存档，如 JAR 或 WAR 文件。

Maven 的一个关键方面是能够找到工件和管理它们之间的依赖关系。*Maven 协调* 是一组用于标识特定工件位置的值。基本协调有三个值，格式为：

groupId:artifactId:version

有时，Maven 通过 *打包值或使用打包* 值和 *分类器* 值增加一个基本的协调。*Maven 协调可以具有以下形式之一：*

```
groupId:artifactId:version
groupId:artifactId:packaging:version
groupId:artifactId:packaging:classifier:version
```

以下是值的描述：

groupdId

定义工件名称的范围。您通常使用所有或部分软件包名称作为组 ID。例如，org.fusesource.example。

artifactId

定义相对于组 ID 的工件名称。

version

指定工件的版本。版本号最多可有四个部分：n.n.n.n，其中版本号的最后一部分可以包含非数字字符。例如，1.0-SNAPSHOT 的最后一部分是字母数字字符子字符串 0-SNAPSHOT。

打包

定义构建项目时生成的打包实体。对于 OSGi 项目，打包是 捆绑包。默认值为 jar。

分类器

可让您区分从同一 POM 构建但具有不同内容的工件。

工件的 POM 文件中的元素定义工件的组 ID、工件 ID、打包和版本，如下所示：

```
<project ... >
  ...
  <groupId>org.fusesource.example</groupId>
  <artifactId>bundle-demo</artifactId>
  <packaging>bundle</packaging>
  <version>1.0-SNAPSHOT</version>
  ...
</project>
```

要定义对上述工件的依赖项，您可以在 POM 文件中添加以下 dependencies 元素：

```
<project ... >
  ...
  <dependencies>
    <dependency>
      <groupId>org.fusesource.example</groupId>
      <artifactId>bundle-demo</artifactId>
      <version>1.0-SNAPSHOT</version>
    </dependency>
  </dependencies>
  ...
</project>
```

注意

不需要在前面的依赖项中指定 bundle 软件包类型，因为捆绑包只是特定类型的 JAR 文件，jar 是默认的 Maven 软件包类型。但是，如果您需要在依赖项中明确指定打包类型，您可以使用 type 元素。