Red Hat JBoss Enterprise Application Platform 8.0

为 JBoss EAP 部署开发应用程序入门

开始为 JBoss EAP 部署创建应用程序。

Last Updated: 2024-02-09

Red Hat JBoss Enterprise Application Platform 8.0 为 JBoss EAP 部署开发应用程序入门

开始为 JBoss EAP 部署创建应用程序。

法律通告

Copyright © 2024 Red Hat, Inc.

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 ® is the registered trademark of Linus Torvalds in the United States and other countries.

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

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

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

Node.js ® 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 ® 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.

摘要

使用 Maven 作为项目管理工具，开始为 JBoss EAP 部署创建应用程序。将您的应用程序部署到 JBoss EAP 在裸机或 OpenShift Container Platform 上运行。

目录

提供有关 JBOSS EAP 文档的反馈

要报告错误或改进文档，请登录到 Red Hat JIRA 帐户并提交问题。如果您没有 Red Hat Jira 帐户，则会提示您创建一个帐户。

流程

1. 单击以下链接 **以创建 ticket**。
2. 在 **Summary** 中输入问题的简短描述。
3. 在 **Description** 中提供问题或功能增强的详细描述。包括一个指向文档中问题的 URL。
4. 点 **Submit** 创建问题，并将问题路由到适当的文档团队。

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。由于此项工作十分艰巨，这些更改将在即将推出的几个发行版本中逐步实施。有关更多详情，请参阅我们的首席技术官 Chris Wright 提供的消息。

熟悉新编程语言或技术的最佳方法是创建"Hello World"应用程序。您可以使用 Maven 作为项目管理工具，为 JBoss EAP 创建"Hello World"应用。

要创建 Hello World 应用程序，请进行部署并测试部署，请按照以下步骤操作：

裸机部署

- 为 hello world 应用创建一个 Maven 项目
- 创建一个 hello world servlet
- 将应用程序部署到裸机安装
- 添加集成测试所需的 Maven 依赖项和配置集
- 测试在裸机上运行的 JBoss EAP 上部署的应用程序

OpenShift Container Platform 部署

- 为 hello world 应用创建一个 Maven 项目
- 创建一个 hello world servlet
- 将应用程序部署到 OpenShift Container Platform
- 添加集成测试所需的 Maven 依赖项和配置集
- 测试部署到 OpenShift Container Platform 上的 JBoss EAP 的应用程序

第 1 章 为 HELLO WORLD 应用创建一个 MAVEN 项目

Maven 项目包含 **pom.xml** 配置文件，具有创建应用所需的目录结构。您可以配置 **pom.xml** 配置文件，以添加应用的依赖项。

要为 hello world 应用程序创建 Maven 项目，请按照以下步骤操作：

- 使用 **maven-archetype-webapp**创建一个 Maven 项目
- 在 Maven 项目中定义属性
- 在 Maven 项目中定义存储库
- 将 JBoss EAP BOM 作为依赖项管理导入到 Maven 项目中
- 在 Maven 项目中添加插件管理
- 验证 maven 项目

1.1. 使用 MAVEN-ARCHETYPE-WEBAPP创建一个 MAVEN 项目

使用 **maven-archetype-webapp** archetype 创建一个 Maven 项目，用于构建用于 JBoss EAP 部署的应用程序。Maven 提供不同的架构类型，用于基于特定于项目类型的模板创建项目。**maven-archetype-webapp** 创建一个项目，其中包含开发简单 web-applications 所需的结构。

先决条件

- 您已安装了 Maven。如需更多信息，请参阅 下载 Apache Maven。

流程

1. 使用 **mvn** 命令设置 Maven 项目。该命令创建项目的目录结构以及 **pom.xml** 配置文件。

```
$ mvn archetype:generate                \
-DgroupId=org.jboss.as.quickstarts      \ 1
-DartifactId=helloworld                 \ 2
-DarchetypeGroupId=org.apache.maven.archetypes  \ 3
-DarchetypeArtifactId=maven-archetype-webapp    \ 4
-DinteractiveMode=false                 5
```

1. **GroupId** 唯一标识项目。
2. **artifactId** 是生成的 **jar** 归档的名称。
3. **maven-archetype-webapp** 的 **groupID**。
4. **maven-archetype-webapp** 的 **artifactID**。
5. 告知 Maven 使用提供的参数，而不是启动交互模式。

2. 导航到生成的目录。

```
$ cd helloworld
```

3. 在文本编辑器中打开生成的 **pom.xml** 配置文件。

4. 在 <name& **gt; helloworld Maven Webapp</name> 行后，删除 pom.xml 配置文件的 <project >** 部分的内容。
确保文件类似如下：

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

<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
   <modelVersion>4.0.0</modelVersion>

   <groupId>org.jboss.as.quickstarts</groupId>
   <artifactId>helloworld</artifactId>
   <version>1.0-SNAPSHOT</version>
   <packaging>war</packaging>
   <name>helloworld Maven Webapp</name>

</project>
```

内容已被删除，因为应用程序不需要它。

后续步骤

- 在 Maven 项目中为 JBoss EAP hello world 应用定义属性 。

1.2. 在 MAVEN 项目中定义属性

您可以在 Maven **pom.xml** 配置文件中定义属性，作为值的所有者。将 JBoss EAP 服务器的值定义为属性，以在配置中一致地使用该值。

先决条件

- 您已初始化了一个 Maven 项目。
如需了解更多信息，请参阅 为 JBoss EAP hello world 应用初始化 Maven 项目 。

流程

- 定义属性 & **lt;version.server** > 作为您要在其上部署配置的应用程序的 JBoss EAP 版本。

```
<project>
   ...
   <properties>
      <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
      <maven.compiler.source>11</maven.compiler.source>
      <maven.compiler.target>11</maven.compiler.target>
      <version.server>8.0.0.GA-redhat-00009</version.server>
   </properties>
</project>
```

后续步骤

- 在 Maven 项目中定义存储库。

1.3. 在 MAVEN 项目中定义存储库

定义工件和插件存储库，Maven 会在其中查找要下载的工件和插件。

先决条件

- 您已初始化了一个 Maven 项目。
  如需了解更多信息，请参阅 为 JBoss EAP hello world 应用初始化 Maven 项目 。

流程

1. 定义工件存储库。

```
<project>
  ...
  <repositories>
    <repository>                                                    (1)
      <id>jboss-public-maven-repository</id>
      <name>JBoss Public Maven Repository</name>
      <url>https://repository.jboss.org/nexus/content/groups/public/</url>
      <releases>
        <enabled>true</enabled>
        <updatePolicy>never</updatePolicy>
      </releases>
      <snapshots>
        <enabled>true</enabled>
        <updatePolicy>never</updatePolicy>
      </snapshots>
      <layout>default</layout>
    </repository>
    <repository>                                                    (2)
      <id>redhat-ga-maven-repository</id>
      <name>Red Hat GA Maven Repository</name>
      <url>https://maven.repository.redhat.com/ga/</url>
      <releases>
        <enabled>true</enabled>
        <updatePolicy>never</updatePolicy>
      </releases>
      <snapshots>
        <enabled>true</enabled>
        <updatePolicy>never</updatePolicy>
      </snapshots>
      <layout>default</layout>
    </repository>
  </repositories>
</project>
```

(1) Red Hat GA Maven 存储库提供所有产品化的 JBoss EAP 和其他红帽工件。

(2) JBoss Public Maven 存储库提供工件，如 WildFly Maven 插件

2. 定义插件存储库。

```
<project>
  ...
  <pluginRepositories>
    <pluginRepository>
      <id>jboss-public-maven-repository</id>
      <name>JBoss Public Maven Repository</name>
      <url>https://repository.jboss.org/nexus/content/groups/public/</url>
      <releases>
        <enabled>true</enabled>
      </releases>
      <snapshots>
        <enabled>true</enabled>
      </snapshots>
    </pluginRepository>
    <pluginRepository>
      <id>redhat-ga-maven-repository</id>
      <name>Red Hat GA Maven Repository</name>
      <url>https://maven.repository.redhat.com/ga/</url>
      <releases>
        <enabled>true</enabled>
      </releases>
      <snapshots>
        <enabled>true</enabled>
      </snapshots>
    </pluginRepository>
  </pluginRepositories>
</project>
```

后续步骤

- 在 Maven 项目中导入 JBoss EAP BOM 依赖关系管理 。

1.4. 将 JBOSS EAP BOM 作为依赖项管理导入到 MAVEN 项目中

导入带有 Tools Bill of materials (BOM)的 JBoss EAP EE，以控制运行时 Maven 依赖项的版本。当您在 < **dependencyManagement&** gt; 部分中指定 BOM 时，您不需要单独指定 **提供** 的范围中定义的 Maven 依赖项版本。

先决条件

- 您已初始化了一个 Maven 项目。
  如需了解更多信息，请参阅 为 JBoss EAP hello world 应用初始化 Maven 项目 。

流程

1. 在 **pom.xml** 配置文件的 properties 部分为 BOM 版本添加属性。

```
<properties>
  ....
  <version.bom.ee>${version.server}</version.bom.ee>
</properties>
```

属性 < **version.server&gt**; 中定义的值用作 BOM 版本的值。

2. 导入 JBoss EAP BOM 依赖关系管理。

```
<project>
  ...
  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>org.jboss.bom</groupId> (1)
        <artifactId>jboss-eap-ee-with-tools</artifactId> (2)
        <version>${version.bom.ee}</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
</project>
```

(1) JBoss EAP 提供的 BOM 的 GroupId。

(2) JBoss EAP 提供的 BOM 的 artifactId，它提供受支持的 JBoss EAP Java EE API，以及额外的 JBoss EAP API JAR 和客户端 BOM 以及 JAAS 等开发工具。

后续步骤

- 在 Maven 项目中添加插件管理

1.5. 在 MAVEN 项目中添加插件管理

将 Maven 插件管理部分添加到 **pom.xml** 配置文件中，以获取 Maven CLI 命令所需的插件。

先决条件

- 您已初始化了一个 Maven 项目。
  如需了解更多信息，请参阅 为 JBoss EAP hello world 应用初始化 Maven 项目 。

流程

1. 在 < **properties** > 部分中，定义 **wildfly-maven-plugin** 和 **maven-war-plugin** 的版本。

```
<properties>
  ...
  <version.plugin.wildfly>4.1.1.Final</version.plugin.wildfly>
  <version.plugin.war>3.3.2</version.plugin.war>
</properties>
```

2. 在 & **lt; project> 部分的 <build& gt; 部分添加 <pluginManagement &** gt;。

```
<project>
  ...
  <build>
    <pluginManagement>
      <plugins>
        <plugin> (1)
```

```
            <groupId>org.wildfly.plugins</groupId>
            <artifactId>wildfly-maven-plugin</artifactId>
            <version>${version.plugin.wildfly}</version>
        </plugin>
        <plugin>                                    2
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-war-plugin</artifactId>
            <version>${version.plugin.war}</version>
        </plugin>
    </plugins>
  </pluginManagement>
 </build>
</project>
```

1. 您可以使用 **wildfly-maven-plugin** 将应用程序部署到 JBoss EAP，使用 **wildfly:deploy** 命令。
2. 您需要管理 war 插件版本，以确保与 JDK17+ 兼容。

后续步骤

- 验证 maven 项目

1.6. 验证 MAVEN 项目

验证您配置的 Maven 项目。

先决条件

- 您已定义了 Maven 属性。
  如需更多信息，请参阅 Maven 项目中的定义属性。
- 您已定义了 Maven 存储库。
  如需更多信息，请参阅在 Maven 项目中定义存储库。
- 您已将 JBoss EAP Bill 材料(BOMs)导入为依赖项管理。
  如需更多信息，请参阅在 Maven 项目中将 JBoss EAP BOM 导入为依赖项管理。
- 您已添加了插件管理。
  如需更多信息，请参阅为服务器 hello world 应用在 Maven 项目中添加插件管理。

流程

- 安装本地添加到 **pom.xml** 中的 Maven 依赖项。

```
$ mvn package
```

您会看到类似如下的输出：

```
...
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
```

```
[INFO] ------------------------------------------------------------------------
...
```

后续步骤

- 创建一个 hello world servlet

第 2 章 创建一个 HELLO WORLD SERVLET

创建一个 servlet，它将在访问时返回 "Hello world!"。

在此过程中，*<application_home>* 指向包含应用程序 **pom.xml** 配置文件的目录。

先决条件

- 您已创建了一个 Maven 项目。
  如需更多信息，请参阅为 Hello World 应用创建一个 Maven 项目 。

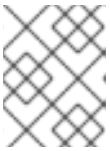
流程

1. 在 < **dependencyManagement** > 部分后，将所需的依赖项添加到 **pom.xml** 配置文件。

```
<project>
  ...
  <dependencies>
    <dependency>                                    (1)
      <groupId>jakarta.servlet</groupId>
      <artifactId>jakarta.servlet-api</artifactId>
      <scope>provided</scope>                       (2)
    </dependency>
  </dependencies>
```

(1) **jakarta.servlet-api** 依赖项提供 Jakarta Servlet API。

(2) 定义 **提供** 的范围，以便依赖项不包含在应用程序中。不包含应用中的依赖项的原因是，此依赖项由 **jboss-eap-ee-with-tools** BOM 管理，此类依赖项包含在 JBoss EAP 中。



注意

依赖项在没有版本的情况下定义，因为 **jboss-eap-ee-with-tools** BOM 在 < **dependencyManagement&** gt; 部分中导入。

2. 进入 < *application_home*> 目录。
3. 创建一个用于存储 Java 文件的目录。

```
$ mkdir -p src/main/java/org/jboss/as/quickstarts/helloworld
```

4. 前往新目录。

```
$ cd src/main/java/org/jboss/as/quickstarts/helloworld
```

5. 创建 return "Hello World!" 的 Servlet **HelloWorldServlet.java**。

```
package org.jboss.as.quickstarts.helloworld;

import java.io.IOException;
import java.io.PrintWriter;
import jakarta.servlet.ServletException;
```

```
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

@WebServlet("/HelloWorld")                    (1)
public class HelloWorldServlet extends HttpServlet {

    static String PAGE_HEADER = "<html><head><title>helloworld</title></head><body>";

    static String PAGE_FOOTER = "</body></html>";

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        resp.setContentType("text/html");
        PrintWriter writer = resp.getWriter();
        writer.println(PAGE_HEADER);
        writer.println("<h1> Hello World! </h1>");
        writer.println(PAGE_FOOTER);
        writer.close();
    }
}
```

(1) **@WebServlet ("/HelloWorld")** 注释为 JBoss EAP 提供以下信息：

- 此类是 servlet。
- 在 URL " <*application_URL*> /HelloWorld" 中提供 servlet。
  例如，如果 JBoss EAP 在 localhost 上运行，且位于默认的 HTTP 端口 8080，则 URL 为 **http://localhost:8080/helloworld/HelloWorld**。

6. 进入 <*application_home*>/src/main/webapp 目录。
   您会找到 Maven 创建的"index.jsp"文件。当您访问应用程序时，此文件会输出 "Hello World!"。
7. 通过用以下内容替换其内容，更新 "index.jsp" 文件，以重定向到 Hello World servlet ：

```
<html>
   <head>
      <meta http-equiv="Refresh" content="0; URL=HelloWorld">
   </head>
</html>
```

8. 进入 <*application_home*> 目录。
9. 使用以下命令编译并打包应用程序作为 Web 归档(WAR)：

```
$ mvn package
```

输出示例

```
...
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
```

```
[INFO] ------------------------------------------------------------------------
...
```

后续步骤

- 将应用部署到服务器

第 3 章 将应用部署到服务器

您可以在裸机或 OpenShift Container Platform 上运行的 JBoss EAP 服务器上部署应用程序。

要在裸机上运行的 JBoss EAP 服务器上部署应用程序，请按照以下步骤执行：

- 将应用程序部署到裸机安装

要在 OpenShift Container Platform 上运行的 JBoss EAP 服务器上部署应用程序，请按照以下步骤执行：

- 准备应用程序以便在 OpenShift Container Platform 上部署
- 使用 Helm 将应用程序部署到 OpenShift 上的 JBoss EAP

3.1. 将应用程序部署到裸机安装

您可以使用 JBoss EAP 部署插件将应用部署到 JBoss EAP。

先决条件

- 您已创建了应用程序。
  如需更多信息，请参阅创建 hello world servlet。
- JBoss EAP 正在运行。

流程

1. 导航到应用程序根目录。
   应用根目录包含 **pom.xml** 配置文件。
2. 将以下构建配置添加到 < **project** > 部分中的 **pom.xml** 配置文件，以定义应用程序存档文件名。

```
<build>
    ...
    <finalName>${project.artifactId}</finalName>      (1)
</build>
```

(1) 将部署的名称设置为项目的工件 ID。

3. 使用 JBoss EAP 部署插件构建和部署应用。

```
$ mvn package wildfly:deploy
```

验证

- 在浏览器中访问地址 **http://localhost:8080/helloworld/**。
  您将被重定向到 http://localhost:8080/helloworld/HelloWorld，您会收到以下信息：

```
Hello World!
```

后续步骤

- 测试在 JBoss EAP 上部署的应用程序

3.2. 将应用程序部署到 OPENSHIFT CONTAINER PLATFORM

您可以使用 Source-to-image (S2I)工作流将应用程序部署到 OpenShift Container Platform 上的 JBoss EAP。S2I 工作流从 Git 存储库获取源代码，并将它注入到基于您要使用的语言和框架的容器中。在 S2I 工作流完成后，会编译 **src** 代码，应用将被打包并部署到 JBoss EAP 服务器。

3.2.1. 准备应用程序以便在 OpenShift Container Platform 上部署

OpenShift Container Platform 使用托管在 Git 存储库中的应用程序。要在 OpenShift 上部署您的应用，您必须首先将应用推送到 Git 存储库。之后，您可以使用 JBoss EAP helm chart 来配置应用程序部署。

先决条件

- 您已创建了应用程序。
  如需更多信息，请参阅创建 Hello World servlet。
- 您已创建了 Git 存储库。

流程

1. 如果应用程序还没有位于其中，请将应用程序移到本地 Git 存储库。

```
$ mv -r helloworld/ <your_git_repo>
```

2. 在 **pom.xml** 配置文件中定义以下属性：

```
<properties>
  ...
  <version.plugin.eap>1.0.0.Final-redhat-00013</version.plugin.eap> (1)
</properties>
```

(1) **<version.plugin.eap** > 定义 JBoss EAP Maven 插件的版本。

3. 将 JBoss EAP maven 插件添加到 < **pluginManagement&** gt;，在 &lt **；project> 部分的 <build &gt**; 部分。

```
<project>
  ...
  <build>
    <pluginManagement>
      <plugins>
        ...
        <plugin>
          <groupId>org.jboss.eap.plugins</groupId>
          <artifactId>eap-maven-plugin</artifactId>
          <version>${version.plugin.eap}</version>
        </plugin>
      </plugins>
    </pluginManagement>
  </build>
</project>
```

4. 在 **pom.xml** 配置文件中创建配置文件"openshift"。
此配置集定义了在 OpenShift Container Platform 上部署所需的插件、功能软件包和层。

```
<profiles>
  <profile>
    <id>openshift</id>
    <build>
      <plugins>
        <plugin>
          <groupId>org.jboss.eap.plugins</groupId>
          <artifactId>eap-maven-plugin</artifactId>
          <configuration>
            <channels>
              <channel>
                <manifest>
                  <groupId>org.jboss.eap.channels</groupId>
                  <artifactId>eap-8.0</artifactId>
                </manifest>
              </channel>
            </channels>
            <feature-packs>
              <feature-pack>
                <location>org.jboss.eap:wildfly-ee-galleon-pack</location>
              </feature-pack>
              <feature-pack>
                <location>org.jboss.eap.cloud:eap-cloud-galleon-pack</location>
              </feature-pack>
            </feature-packs>
            <layers>
              <layer>cloud-server</layer>
            </layers>
            <name>ROOT.war</name>
          </configuration>
          <executions>
            <execution>
              <goals>
                <goal>package</goal>
              </goals>
            </execution>
          </executions>
        </plugin>
      </plugins>
    </build>
  </profile>
</profiles>
```

1. **wildfly-maven-plugin** 是一个 JBoss EAP 插件，用于在 OpenShift Container Platform 上部署应用程序置备 JBoss EAP 实例。

2. **功能包** 定义了功能包（包含用于动态置备服务器的功能的文件）。在这种情况下，我们需要 feature-packs **org.wildfly:wildfly-galleon-pack** 和 **org.wildfly.cloud:wildfly-cloud-galleon-pack**。

3.

层 定义要在调配的服务器中包含的层（来自配置的功能包）。每个层标识了一个或多个可以自行安装的服务器功能，或者与其他层结合使用。在我们的情形中，我们选择了 **cloud-**

4 **<name>ROOT.war&** lt;/name> ：定义应用程序 web 归档(WAR)的结果名称。如果指定了 **ROOT.war**，则应用程序会在服务器的 root 路径上部署，否则会在 < **name/>** 相对路径上部署。

5. 验证应用程序是否编译。

```
$ mvn package -Popenshift
```

6. 将更改推送到您的存储库。

后续步骤

- 使用 Helm 将应用程序部署到 OpenShift 上的 JBoss EAP

3.2.2. 使用 Helm 将应用程序部署到 OpenShift 上的 JBoss EAP

使用 JBoss EAP Helm Chart 使用 Helm 配置和部署应用程序到 OpenShift 上的 JBoss EAP。

先决条件

- 您已准备了应用程序以在 OpenShift Container Platform 上部署。
  如需更多信息，请参阅准备应用程序以便在 OpenShift Container Platform 上部署。
- 您已在 OpenShift Container Platform 中创建了一个项目。
  如需更多信息，请参阅使用项目。
- 已安装 OpenShift CLI (**oc**)
  如需更多信息，请参阅安装 OpenShift CLI。
- 从您的机器登录到 OpenShift Container Platform。
  如需更多信息，请参阅 OpenShift CLI 登录。
- 已安装 helm。
  如需更多信息，请参阅安装 Helm。

流程

1. 在应用程序 root directoy 中创建一个名为 **charts** 的目录并导航到它。应用根目录是包含 **pom.xml** 配置文件的一个目录。

```
$ mkdir charts; cd charts
```

2. 使用以下内容创建文件 **helm.yaml** ：

```
build:
  uri: https://github.com/<user>/<repository>.git   1
  ref: <branch_name>                                2
  contextDir: helloworld                            3
deploy:
  replicas: 1                                       4
```

```
-
```

1. 指定包含要在 OpenShift Container Platform 上部署的应用程序的 Git 存储库的 URL。
2. 指定包含应用程序的 Git 分支。
3. 指定包含应用程序的目录。
4. 指定要创建的 pod 数量。

3. 在 Helm 中配置 JBoss EAP 仓库。
   - 如果您之前没有将 JBoss EAP 仓库添加到 Helm，请添加它。

     ```
     $ helm repo add jboss-eap https://jbossas.github.io/eap-charts/
     ```

   - 如果您已将 JBoss EAP 存储库添加到 Helm，请更新它。

     ```
     $ helm repo update jboss-eap
     ```

4. 使用 helm 部署应用程序。

   ```
   $ helm install helloworld -f helm.yaml jboss-eap/eap8
   ```

   完成部署可能需要几分钟时间。

验证

1. 获取指向部署的路由的 URL。

   ```
   $ APPLICATION_URL=https://$(oc get route helloworld --template='{{ .spec.host }}') &&
   echo "" &&
   echo "Application URL: $APPLICATION_URL"
   ```

2. 在浏览器中导航至"应用程序 URL"。
   您会在路径 "/HelloWorld" 重定向到 servlet，您会收到以下消息：

   ```
   Hello World!
   ```

后续步骤

- 测试在 JBoss EAP 上部署的应用程序

第 4 章 测试在 JBOSS EAP 上部署的应用程序

为确保 JBoss EAP 上部署的 Hello World 应用正常工作，您可以添加集成测试。

要为在裸机上运行的 JBoss EAP 服务器上部署的应用程序添加测试，请按照以下步骤操作：

- 添加集成测试所需的 Maven 依赖项和配置集
- 创建测试类来测试应用程序
- 测试在裸机上运行的 JBoss EAP 上部署的应用程序

要为在 OpenShift Container Platform 上运行的 JBoss EAP 服务器上部署的应用程序添加测试，请按照以下步骤执行：

- 添加集成测试所需的 Maven 依赖项和配置集
- 创建测试类来测试应用程序
- 测试部署到 OpenShift Container Platform 上的 JBoss EAP 的应用程序

4.1. 添加集成测试所需的 MAVEN 依赖项和配置集

要为应用程序创建集成测试，请添加所需的 Maven 依赖项。

先决条件

- 您已创建了一个 Maven 项目。
  如需更多信息，请参阅为 hello world 应用创建一个 Maven 项目 。

流程

1. 在 **pom.xml** 配置文件中定义以下属性：

```
<properties>
    ...
    <version.plugin.failsafe>3.2.2</version.plugin.failsafe>
</properties>
```

2. 添加测试所需的依赖项。

```
<project>
    ...
    <dependencies>
        ...
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>
</project>
```

3. 定义一个配置文件，以添加集成测试所需的插件。

```
<project>
  ...
  <profiles>
  ...
    <profile>
      <id>integration-testing</id>
      <build>
        <plugins>
          <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-failsafe-plugin</artifactId> (1)
            <version>${version.plugin.failsafe}</version>
            <configuration>
              <includes>
                <include>**/HelloWorldServletIT</include> (2)
              </includes>
            </configuration>
            <executions>
              <execution>
                <goals>
                  <goal>integration-test</goal>
                  <goal>verify</goal>
                </goals>
              </execution>
            </executions>
          </plugin>
        </plugins>
      </build>
    </profile>
  </profiles>
</project>
```

1. 用于运行集成测试的 Maven 插件。
2. 测试应用的 Java 类的名称。

后续步骤

- 创建测试类来测试应用程序

4.2. 创建测试类来测试应用程序

通过检查 Web 页面的 HTTP GET 是否返回 200 OK，创建一个集成测试，验证应用程序是否已在 OpenShift Container Platform 上的 JBoss EAP 上部署并在运行。

在此过程中，*<application_home>* 指向包含应用程序 **pom.xml** 配置文件的目录。

先决条件

- 您已将应用程序部署到 JBoss EAP。
  如需更多信息，请参阅 构建和部署到服务器。
- 您已添加了 JUnit 测试所需的 Maven 依赖项。
  如需更多信息，请参阅 为集成测试所需的 Maven 依赖项和配置集。

流程

1. 进入 *<application_home>* 目录。
2. 创建用于存储测试类的目录。

```
$ mkdir -p src/test/java/org/jboss/as/quickstarts/helloworld
```

3. 前往新目录。

```
$ cd src/test/java/org/jboss/as/quickstarts/helloworld
```

4. 创建一个 Java 类 **HelloWorldServletIT.java** 来测试部署。

```
package org.jboss.as.quickstarts.helloworld;

import org.junit.Test;
import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.time.Duration;
import static org.junit.Assert.assertEquals;

public class HelloWorldServletIT {

    private static final String DEFAULT_SERVER_HOST = "http://localhost:8080/helloworld";
(1)

    @Test
    public void testHTTPEndpointIsAvailable() throws IOException, InterruptedException,
URISyntaxException {
        String serverHost = System.getProperty("server.host");
        if (serverHost == null) {
            serverHost = DEFAULT_SERVER_HOST;
        }
        final HttpRequest request = HttpRequest.newBuilder()
                .uri(new URI(serverHost+"/HelloWorld"))
                .GET()
                .build();                                                    (2)
        final HttpClient client = HttpClient.newBuilder()
                .followRedirects(HttpClient.Redirect.ALWAYS)
                .connectTimeout(Duration.ofMinutes(1))
                .build();                                                    (3)
        final HttpResponse<String> response = client.send(request,
HttpResponse.BodyHandlers.ofString()); (4)
        assertEquals(200, response.statusCode());                                    (5)
    }
}
```

(1) 运行应用程序的 URL。如果 **sever.host** 未定义，则使用这个值。

2. 为应用 URI 创建 HttpRequest 实例。
3. 创建一个 HttpClient 来发送请求，并从应用接收响应。
4. 从应用程序获取响应。
5. 测试从应用程序重新邀请的响应是否为"200"，表示应用程序可以被重新使用。

后续步骤

- 要测试在裸机上运行的 JBoss EAP 服务器上部署的应用程序，请按照以下步骤执行：
  - 测试在裸机上运行的 JBoss EAP 上部署的应用程序
- 要测试部署在 OpenShift Container Platform 上运行的 JBoss EAP 服务器上的应用程序，请按照以下步骤执行：
  - 测试部署到 OpenShift Container Platform 上的 JBoss EAP 的应用程序

4.3. 测试在裸机上运行的 JBOSS EAP 上部署的应用程序

测试部署在裸机上运行的 JBoss EAP 上的应用。

先决条件

- 您已创建了测试类。
  如需更多信息，请参阅创建测试类来测试应用程序
- 要测试的应用程序部署在 JBoss EAP 上。
- JBoss EAP 正在运行。

流程

1. 进入 <*application_home*> 目录。
2. 使用带有 **integration-testing** 配置集的 **verify** 命令运行集成测试。

```
$ mvn verify -Pintegration-testing
```

输出示例

```
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- maven-failsafe-plugin:3.2.2:verify (default) @ helloworld ---
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
```

```
[INFO] Total time:  9.982 s
[INFO] Finished at: 2023-11-22T14:53:54+05:30
[INFO] ------------------------------------------------------------------------
```

4.4. 测试部署到 OPENSHIFT CONTAINER PLATFORM 上的 JBOSS EAP 的应用程序

测试部署到 OpenShift Container Platform 上的 JBoss EAP 的应用程序。

先决条件

- 您已创建了测试类。
  如需更多信息，请参阅创建测试类来测试应用程序

流程

1. 将更改推送到您的 Git 存储库。
2. 进入 < *application_home*> 目录。
3. 使用 **verify** 命令运行测试，激活 **integration-testing** 配置集并指定应用程序的 URL。

```
$ mvn verify -Pintegration-testing -Dserver.host=https://$(oc get route helloworld --
template='{{ .spec.host }}')
```

注意

测试使用 SSL/TLS 连接到部署的应用程序。因此，您需要运行测试的机器信任证书。

要信任证书，您必须将其添加到 Java 信任存储中。

Example

```
$ keytool -trustcacerts -keystore _<path-to-java-truststore>_ -storepass
_<trust-store-password>_ -importcert -alias _<alias-for-the-certificate>_ -file
_<path-to-certificate>_/_<certificate-name>_
```

输出示例

```
[INFO] Running org.jboss.as.quickstarts.helloworld.HelloWorldServletIT
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 1.345 s -- in
org.jboss.as.quickstarts.helloworld.HelloWorldServletIT
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- maven-failsafe-plugin:3.2.2:verify (default) @ helloworld ---
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
```

```
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  2.984 s
[INFO] Finished at: 2023-11-30T15:51:22+05:30
[INFO] ------------------------------------------------------------------------
```

更新于 2024-02-08