Red Hat JBoss Enterprise Application Platform 7.4

适用于 OpenShift 的 JBoss EAP 在线版入门

为 OpenShift Online 使用红帽 JBoss 企业应用平台进行开发指南

Last Updated: 2024-02-10

Red Hat JBoss Enterprise Application Platform 7.4 适用于 OpenShift 的 JBoss EAP 在线版入门

为 OpenShift Online 使用红帽 JBoss 企业应用平台进行开发指南

法律通告

摘要

在 OpenShift Online 中使用红帽 JBoss 企业应用平台指南

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. 请包含 **文档 URL、章节编号 并描述问题**。
3. 在 **Summary** 中输入问题的简短描述。
4. 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。
5. 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

红帽承诺替换我们的代码、文档和网页属性中存在问题的语言。我们从这四个术语开始： master、slave、blacklist 和 whitelist。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 CTO Chris Wright 信息。

第 1 章 简介

1.1. 什么是红帽 JBOSS 企业应用平台(JBOSS EAP)？

红帽 JBoss 企业应用平台 7.4(JBoss EAP)是基于开放标准构建的中间件平台，符合 Jakarta EE 8 规格。它为高可用性群集、消息传递和分布式缓存等功能提供了预配置选项。它包含一个模块化结构，允许您仅在需要时启用服务，从而提高启动速度。

基于 Web 的管理控制台和管理命令行界面(CLI)无需编辑 XML 配置文件，还增加了编写脚本和自动执行任务的功能。此外，JBoss EAP 还包括 API 和开发框架，使您能够快速开发、部署和运行安全且可扩展的 Jakarta EE 应用。JBoss EAP 7.4 是一种适用于 Web 配置文件和完整平台规格的 Jakarta EE 8 兼容实施。

1.2. JBOSS EAP 如何在 OPENSHIFT 上工作？

红帽为 JBoss EAP 提供容器化映像，专为与 OpenShift 配合使用而设计。开发人员可以利用此镜像快速轻松地构建、扩展和测试跨混合环境部署的应用程序。

1.3. 比较：用于 OPENSHIFT 的 JBOSS EAP 和 JBOSS EAP

将 JBoss EAP 产品与用于 OpenShift 镜像的 JBoss EAP 进行比较时有一些显著的差异。下表描述了这些差异，并记录了适用于 OpenShift 的当前 JBoss EAP 版本中包含的或支持哪些功能。

表 1.1. 用于 OpenShift 的 JBoss EAP 和 JBoss EAP 之间的差异

JBoss EAP 功能	OpenShift 的 JBoss EAP 状态	描述
JBoss EAP 管理控制台	未包括	用于 OpenShift 的 JBoss EAP 管理控制台不包含 JBoss EAP 管理控制台。
JBoss EAP 管理 CLI	不推荐	不建议将 JBoss EAP 管理 CLI 用于容器化环境中运行的 JBoss EAP。当容器重启时，使用管理 CLI 在运行的容器中进行的任何配置更改都将丢失。管理 CLI 可以从容器集内访问，以满足故障排除需要。
受管域	不支持	虽然 JBoss EAP 受管域不受支持，但应用的创建和分发在 OpenShift 上的容器中进行管理。
默认根页面	Disabled	默认根页面已被禁用，但您可以将自己的应用部署至 root 上下文，即 **ROOT.war**。
远程消息传递	支持	支持用于 pod 间和远程消息传递的红帽 AMQ。ActiveMQ Artemis 仅支持具有 JBoss EAP 实例的单一 pod 中的消息传递，并且仅在缺少红帽 AMQ 时启用。

JBoss EAP 功能	OpenShift 的 JBoss EAP 状态	描述
事务恢复	部分支持	EAP 操作器是 OpenShift 4 中唯一经过测试且受支持的事务恢复选项。如需有关使用 EAP 操作器恢复事务的更多信息，请参阅 EAP Operator for Safe Transaction Recovery。 不支持某些场景。有关不支持的场景的更多信息，请参阅 不支持的事务恢复方案。
嵌入式消息传递代理	已弃用	在 OpenShift 容器中使用嵌入式消息传递代理已弃用。在以后的发行版本中将删除对嵌入式代理的支持。 如果容器配置为使用嵌入式消息传递代理，如果没有配置远程代理，则会记录警告信息。 如果容器配置不包括消息传递目的地，请将 **DISABLE_EMBEDDED_JMS_BROKER** 环境变量设置为 **true**，以禁用配置嵌入式消息传递代理的功能。

1.4. 版本兼容性和支持

用于 OpenShift 的 JBoss EAP 为 OpenJDK 8 和 OpenJDK 11 提供镜像。

每个镜像有两种不同的版本：S2I 构建器镜像和运行时镜像。S2I 构建器镜像包含完整的 JBoss EAP 服务器，以及 S2I 构建期间所需的工具。运行时镜像包含运行 JBoss EAP 但不包含服务器所需的依赖项。服务器在连锁构建期间安装在运行时镜像中。

以下修改已应用到 OpenShift 的 JBoss EAP 7.4 中的镜像。

- 已删除默认驱动程序和模块。
- MySQL 和 PostgreSQL 的模板已被删除。您可以使用自定义层置备这些功能。
- 在这些镜像中，Hawkular 代理没有处于活动状态。如果配置，则忽略它。
- 默认情况下，容器启动时不再添加数据源 **ExampleDS**。如果您需要默认数据源，请使用环境变量 **ENABLE_GENERATE_DEFAULT_DATASOURCE**，其值为 **true** (**ENABLE_GENERATE_DEFAULT_DATASOURCE=true**)

注意

以下发现机制协议已弃用，并被其他协议替代：

- **openshift.DNS_PING** 协议已弃用，并替换为 **dns.DNS_PING** 协议。如果您在自定义的 **standalone-openshift .xml** 文件中引用了 **openshift. DNS_PING** 协议，请将协议替换为 **dns.DNS_PING** 协议。
- **openshift.KUBE_PING** 发现机制协议已弃用，并替换为 **kubernetes.KUBE_PING** 协议。

OpenJDK 镜像支持的构架
OpenJDK 镜像支持多种架构。下表总结了这些信息：

1. OpenJDK 镜像和架构

JDK (OS)	支持的构架	Red Hat Ecosystem Catalog
OpenJDK8(RHEL 7)	x86_64	构建器镜像和 运行时镜像
OpenJDK11 (RHEL 8)	x86_64、IBM Z 和 IBM Power 系统	构建器镜像和 运行时镜像

适用于 OpenShift 的 JBoss EAP 经常更新。因此，务必要了解哪些镜像版本与哪些版本的 OpenShift 兼容。

其他资源

- OpenShift 和 Atomic Platform 测试的集成
- 针对 OpenShift 的 JBoss EAP 的功能调试

1.4.1. OpenShift 4.x 支持

OpenShift 4.1 的更改会影响对 Jolokia 的访问，Open Java 控制台在 OpenShift 4.x Web 控制台中不再可用。

在以前的版本中，在 OpenShift 发行本中，某些 kube-apiserver 代理请求已经过身份验证，并传递给集群。现在，这种行为被视为不安全，因此不再支持以这种方式访问 Jolokia。

由于 OpenShift 控制台代码库更改，Open Java 控制台链接不再可用。

1.4.2. IBM Z 支持

libartemis-native 的 s390x 变体不包含在镜像中。因此，任何与 AIO 相关的设置都不会被考虑。

- **journal-type** ： **将 journal-type** 设置为 **ASYNCIO** 无效。此属性的值在运行时默认为 **NIO**。
- **journal-max-io** ：此属性无效。
- **journal-store-enable-async-io** ：此属性无效。

1.4.3. 在 OpenShift 上从 JBoss EAP 7.1 升级到 JBoss EAP 7.4

OpenShift 上的 JBoss EAP 7.1 安装的 **standalone-openshift.xml** 文件与 JBoss EAP 7.4 及更高版本不兼容。您必须修改随 JBoss EAP 7.1 安装的 **standalone-openshift.xml** 文件，然后使用该文件为 OpenShift 启动 JBoss EAP 7.4 或更高版本容器。

其他资源

在 OpenShift 上将 JBoss EAP 7.1 升级到 JBoss EAP 7.4 时对 **standalone-openshift.xml** 的更新

1.5. 部署选项

您可以使用以下选项之一在 OpenShift 中部署 JBoss EAP Java 应用程序：

- 适用于 OpenShift 的 JBoss EAP 模板.
- EAP 操作器是特定于 JBoss EAP 的控制器，扩展 OpenShift API，以代表 OpenShift 用户创建、配置和管理复杂有状态应用的实例。

注意

EAP 操作器仅支持 OpenShift 4 及更高版本。

其他资源

- 如需有关用于 OpenShift 模板的 JBoss EAP 的更多信息，请参阅 *Git Hub* 上的 jboss-eap-openshift-templates。
- 如需有关 EAP 操作器的更多信息，请参阅 OpenShift 上自动化应用部署的 EAP Operator。

第 2 章 在用于 OPENSHIFT 镜像的 JBOSS EAP 上构建并运行 JAVA 应用

以下工作流演示了如何使用 Source-to-Image(S2I)流程在 JBoss EAP for OpenShift 镜像上构建和运行 Java 应用。

例如，这个过程 **中使用了 Kitchensink** faststart。它使用 Jakarta Server Faces、Jakarta Contexts 和 Dependency Injection、Jkarta 企业 Beans、Jakarta Enterprise Beans、Jakarta Persistence 和 Jakarta Bean Validation 来演示一个支持 Web 的数据库应用。如需更多信息，请参阅 JBoss EAP 7 附带的 BIOSchen **sink** 快速入门。

2.1. 先决条件

此工作流假定您已有有效的 OpenShift Online 订阅，并且您已安装了 OpenShift CLI。

2.2. 为应用部署准备 OPENSHIFT

1. 使用 **oc login 命令登录** 您的 OpenShift 实例。
2. 在 OpenShift 中创建新项目：
   项目允许一组用户组织和管理与其他组不同的内容。您可以使用以下命令在 OpenShift 中创建项目：

   ```
   $ oc new-project <project_name>
   ```

   例如，对于 **kitchensink** quickstart，使用以下命令创建名为 **eap-demo** 的新项目：

   ```
   $ oc new-project eap-demo
   ```

3. **可选** ：创建密钥存储和 secret。

   注意

   如果您使用 OpenShift 项目中支持 HTTPS 的功能，则需要创建密钥存储和机密。例如，如果您使用 **eap74-https-s2i** 模板，则必须创建密钥存储和 secret。

   这个 toolschen **sink** quickstart 的工作流演示不使用 HTTPS 模板，因此不需要密钥存储和 secret。

   a. 创建密钥存储.

   警告

   以下命令将生成自签名证书，但对于生产环境，红帽建议您使用从认证认证机构(CA)购买的 SSL 证书进行 SSL 加密连接(HTTPS)。

   您可以使用 Java **keytool** 命令生成密钥存储：

```
$ keytool -genkey -keyalg RSA -alias <alias_name> -keystore <keystore_filename.jks> -validity 360 -keysize 2048
```

例如，对于 **kitchensink** quickstart，使用以下命令生成密钥存储：

```
$ keytool -genkey -keyalg RSA -alias eapdemo-selfsigned -keystore keystore.jks -validity 360 -keysize 2048
```

b. 从密钥存储创建机密。
使用以下命令，从前面创建的密钥存储创建机密。

```
$ oc create secret generic <secret_name> --from-file=<keystore_filename.jks>
```

例如，对于 **Kitchensink** quickstart，使用以下命令来创建 secret：

```
$ oc create secret generic eap7-app-secret --from-file=keystore.jks
```

2.3. 导入适用于 OPENSHIFT 镜像流和模板的最新 JBOSS EAP

您必须将适用于 OpenShift 的最新 JBoss EAP 镜像流和模板导入到 OpenShift 项目的命名空间中。

注意

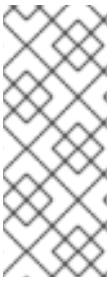
使用您的客户门户网站凭证登录红帽容器注册表，以导入 JBoss EAP 镜像流和模板。如需更多信息，请参阅 Red Hat Container Registry 身份验证 。

为 **JDK 8 导入命令**

```
oc replace -f \
https://raw.githubusercontent.com/jboss-container-images/jboss-eap-openshift-templates/eap74/eap74-openjdk8-image-stream.json
```

此命令导入以下镜像流和模板：

- JDK 8 构建器镜像流： **jboss-eap74-openjdk8-openshift**
- JDK 8 运行时镜像流： **jboss-eap74-openjdk8-runtime-openshift**



注意

如果首次使用 OpenShift 3 并创建 EAP 7.4 ImageStream，请运行以下命令而不是 **oc replace** ：

```
oc create -f https://raw.githubusercontent.com/jboss-container-images/jboss-eap-openshift-templates/eap74/eap74-openjdk8-image-stream.json
```

JDK 11 导入命令

```
oc replace -f \
https://raw.githubusercontent.com/jboss-container-images/jboss-eap-openshift-templates/eap74/eap74-openjdk11-image-stream.json
```

此命令导入以下镜像流和模板：

- JDK 11 构建器镜像流： **jboss-eap74-openjdk11-openshift**
- JDK 11 运行时镜像流： **jboss-eap74-openjdk11-runtime-openshift**

为**模板**导**入命令**

```
for resource in \
  eap74-amq-persistent-s2i.json \
  eap74-amq-s2i.json \
  eap74-basic-s2i.json \
  eap74-https-s2i.json \
  eap74-sso-s2i.json

do
  oc replace -f \
https://raw.githubusercontent.com/jboss-container-images/jboss-eap-openshift-
templates/eap74/templates/${resource}
done
```

此命令导入 命令中指定的所有模板。

注意

使用这些命令导入的 JBoss EAP 镜像流和模板仅在该 OpenShift 项目中可用。

如果要将镜像流和模板导入到其他项目中，请将 **-n *PROJECT_NAME*** 添加到 **命令的 oc replace** 行。例如：

```
...
oc replace -n PROJECT_NAME --force -f
...
```

如果使用 cluster-samples-operator，请参阅有关配置集群样本操作器的 OpenShift 文档。有关配置集群样本操作器的详情，请参阅配置 Samples Operator。

2.4. 将 JBOSS EAP SOURCE-TO-IMAGE(S2I)应用部署到 OPENSHIFT

在导入镜像和模板后，您可以将应用部署到 OpenShift。

先决条件

可选 ：模板可以为许多模板参数指定默认值，您可能需要覆盖部分或全部默认值。要查看模板信息，包括参数列表和任何默认值，请使用命令 **oc describe template *TEMPLATE_NAME***。

流程

1. 创建一个新的 OpenShift 应用，它将 JBoss EAP 用于 OpenShift 镜像和您的 Java 应用的源代码。您可以将其中一个提供的 JBoss EAP 用于 OpenShift 模板，以进行 S2I 构建。您也可以选择调配修剪的服务器。
   例如，若要使用 JDK 8 构建器镜像部署 **kitchensink** 快速启动，请输入以下命令在 **eap-demo** 项**目中使用 eap74-basic -s2i 模板（在 eap-demo** 项目中创建 的 eap **chensink** 源代码）。这个快速入门不支持修剪功能。

```
oc new-app --template=eap74-basic-s2i \ 1
 -p IMAGE_STREAM_NAMESPACE=eap-demo \ 2
 -p EAP_IMAGE_NAME=jboss-eap74-openjdk8-openshift:7.4.0 \ 3
 -p EAP_RUNTIME_IMAGE_NAME=jboss-eap74-openjdk8-runtime-openshift:7.4.0 \ 4
 -p SOURCE_REPOSITORY_URL=https://github.com/jboss-developer/jboss-eap-quickstarts
\ 5
 -p SOURCE_REPOSITORY_REF=7.4.x \ 6
 -p CONTEXT_DIR=kitchensink 7
```

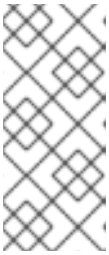
1. 要使用的模板。
2. 最新的镜像流和模板导入到项目命名空间中，因此您必须指定要查找镜像流的命名空间。这通常是项目的名称。
3. JDK8 的 EAP 构建器镜像流的名称。
4. JDK8 的 EAP 运行时镜像流的名称。
5. 包含应用源代码的存储库的 URL。
6. 用于源代码的 Git 存储库引用。这可以是 Git 分支或标签引用。
7. 要构建的源存储库中的目录。

再举一个例子，若要使用 JDK 11 运行时镜像部署 **helloworld-html5** 快速启动并修剪 JBoss EAP 以仅包含 **jaxrs-server** 层，请输入以下命令：命令使用 eap- **demo 项目中的 eap74-basic-s2i** 模板，该模板在为应用部署 准备 OpenShift 中创建， 在 GitHub **上使用 helloworld-html5** 源代码。

```
oc new-app --template=eap74-basic-s2i \ 1
 -p IMAGE_STREAM_NAMESPACE=eap-demo \ 2
 -p EAP_IMAGE_NAME=jboss-eap74-openjdk11-openshift:7.4.0 \ 3
 -p EAP_RUNTIME_IMAGE_NAME=jboss-eap74-openjdk11-runtime-openshift:7.4.0 \ 4
 -p SOURCE_REPOSITORY_URL=https://github.com/jboss-developer/jboss-eap-quickstarts
\ 5
 -p SOURCE_REPOSITORY_REF=7.4.x \ 6
 -p GALLEON_PROVISION_LAYERS=jaxrs-server \ 7
 -p CONTEXT_DIR=helloworld-html5 8
```

1. 要使用的模板。
2. 最新的镜像流和模板导入到项目命名空间中，因此您必须指定要查找镜像流的命名空间。这通常是项目的名称。
3. JDK11 的 EAP 构建器镜像流的名称。
4. JDK11 的 EAP 运行时镜像流的名称。
5. 包含应用源代码的存储库的 URL。
6. 用于源代码的 Git 存储库引用。这可以是 Git 分支或标签引用。
7. 调配仅包含 **jaxrs-server** 层的修剪服务器。

8 要构建的源存储库中的目录。



注意

在创建新的 OpenShift 应用时，您可能还想配置环境变量。

例如，如果您使用 **eap74-https-s2i** 等 HTTPS 模板，您必须指定所需的 HTTPS 环境变量 HTTPS **_NAME、HTTPS_ PASSWORD 和 HTTPS_ KEYSTORE**，才能匹配您的密钥存储详细信息。

注意

如果模板使用 AMQ，则必须包含带有适当值的 **AMQ_IMAGE_NAME** 参数。

如果模板使用 SSO，您必须包含带有适当值的 **SSO_IMAGE_NAME** 参数。

2. 检索构建配置的名称：

```
$ oc get bc -o name
```

3. 使用上一步中的构建配置名称来查看构建的 Maven 进度。

```
$ oc logs -f buildconfig/BUILD_CONFIG_NAME
```

例如，对于 **kitchensink** quickstart，以下命令显示 Maven 构建的进度：

```
$ oc logs -f buildconfig/eap-app
```

其它资源

针对 OpenShift 的 JBoss EAP 的功能调试

2.5. 部署后任务

根据您的应用，可能需要在构建和部署 OpenShift 应用后执行一些任务。这可能包括公开服务，以便可以从 OpenShift 外部查看应用，或者将应用扩展到特定数量的副本。

1. 使用以下命令获取应用的服务名称：

```
$ oc get service
```

2. 将主服务作为路由公开，以便您可以从 OpenShift 外部访问您的应用。例如，对于 **Kitchensink** quickstart，使用以下命令来公开所需的服务和端口：

```
$ oc expose service/eap-app --port=8080
```

注意

如果您使用模板来创建应用，则路由可能已存在。如果存在，请继续下一步。

3. 获取路由的 URL。

```
$ oc get route
```

4. 使用 URL 访问 Web 浏览器中的应用。URL 是上一命令输出中 **HOST/PORT** 字段的值。如果您的应用不使用 JBoss EAP 根上下文，请将应用的上下文附加到 URL。例如，对于 **kitchensink** quickstart，URL 可以是 **http://*HOST_PORT_VALUE*/kitchensink/**。

5. 此外，您还可以通过运行以下命令来扩展应用实例：这会将副本数量增加到 **3**。

```
$ oc scale deploymentconfig DEPLOYMENTCONFIG_NAME --replicas=3
```

例如，对于 **Kitchensink** quickstart，使用以下命令来扩展应用：

```
$ oc scale deploymentconfig eap-app --replicas=3
```

2.6. 在 JBOSS EAP 中为 OPENSHIFT 串联构建支持

用于 OpenShift 的 JBoss EAP 支持 OpenShift 中的连锁构建。

用于 OpenShift 模板的 JBoss EAP 采用连锁构建。当您使用这些模板时，两个构建结果：

- 名为 **[application name]-build-artifacts**的中间镜像
- 最终镜像 **[应用程序名称]**

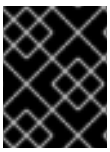
有关连锁构建的详情，请参阅 OpenShift 文档。

其它资源

OpenShift Chained 构建文档

第 3 章 使用 HELM CHART 在 OPENSHIFT 上部署 JBOSS EAP 7 应用程序

您可以使用 Helm chart 在 OpenShift 上使用 JBoss EAP 7 部署并运行 Jakarta EE 应用程序。Helm 是一个软件包管理程序，它简化了应用程序和服务部署到 OpenShift Container Platform 集群的过程。Helm 使用名为 chart 的打包格式。Helm chart 是描述 OpenShift Container Platform 资源的一个文件集合。以下流程演示了如何在 OpenShift Container Platform Web 控制台中使用 Helm chart 部署和运行 Jakarta EE 应用程序。

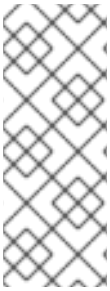


重要

此功能仅作为技术预览提供。在生产环境中不支持使用它，并可能会在以后更改。如需有关技术预览功能支持范围的信息，请参阅红帽客户门户网站上的 技术预览功能支持范围。

3.1. 先决条件

- 已安装并运行 OpenShift 实例。如需有关安装和配置 OpenShift 实例的更多信息，请参阅 OpenShift Container Platform 入门指南。
- 已登陆到 OpenShift Container Platform Web 控制台。有关使用 OpenShift Web 控制台的更多信息，请参阅 OpenShift Container Platform 入门指南。



注意

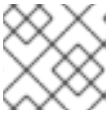
- 您还可以使用 OpenShift Sandbox 在 OpenShift Container Platform 上部署并运行 JBoss EAP 应用程序。这是一个试用沙盒，可在有限的时间内提供。
- 本文档使用 Jakarta EE 应用程序示例。您可以使用相同的步骤来部署自己的 Jakarta EE 应用。如需更多信息，请参阅 https://github.com/jboss-eap-up-and-running/eap7-getting-started

3.2. 使用 HELM 创建 JBOSS EAP 7 应用程序

您可以使用 OpenShift Web 控制台中的 Helm Chart 创建 JBoss EAP 7 应用程序。

流程

1. 在主导航中，点下拉菜单并选择 **Developer**。
2. 在导航菜单中点 **Add**。
   此时会打开 **Add** 页面。
3. 在 **Add** 页面中，点 **Helm Chart**。
4. 在 **Helm Charts** 目录中，搜索 **JBoss EAP 7.4**。
5. 点 **JBoss EAP 7.4** Helm Chart 标题。
   侧面板显示有关 JBoss EAP 7 Helm Chart 的信息。
6. 点 **Install Helm Chart**。
   一些表单部分会被默认折叠。点 **&** gt; 扩展并查看其内容。



注意

这些部分不需要更新才能继续。

在 **build.uri** 字段中指定有关您要构建和部署的 Jakarta EE 应用程序的详细信息。

```
build:
  uri: https://github.com/jboss-eap-up-and-running/eap7-getting-started
```

注意

如果要构建不同的应用程序，您必须更改此 **uri** 字段以指向该应用的 Git 存储库。

7. 点 **Install** 使用 Helm Chart 创建 JBoss EAP 7 应用程序。

验证

- Helm 发行版本由一个短划线框表示，其中包含 JBoss EAP 图标和 **eap74** 文本。此内容放置在短划线框之外。部署通过短划线框中带有文本 **D eap74** 的圆圈来表示。
  - 验证您是否看到 **eap74** Helm 发行版本。
  - 验证您是否看到 **eap74** 部署。

3.3. 查看 HELM 发行版本

使用 Helm Chart 成功创建 JBoss EAP 7 应用程序后，您可以查看与 Helm 发行版本相关的所有信息。

先决条件

- 已使用 Helm Chart 创建 JBoss EAP 7 应用程序。请参阅使用 Helm 创建 JBoss EAP 7 应用程序。

流程

1. 在导航菜单中点 **Helm**。
2. 点 **eap74** Helm release。
   Helm Release 详情页面 将打开。它显示与您安装 Helm 发行版本相关的所有信息。
3. 单击 **Resources** 选项卡。它列出了此 Helm 发行版本创建的所有资源。

验证

- 验证您看到 Helm release **eap74** 旁边的 **Deployed** 标签。

3.4. 查看关联的代码

使用 Helm Chart 成功创建 JBoss EAP 7 应用程序后，您可以查看关联的代码。

先决条件

- 已使用 Helm Chart 创建 JBoss EAP 7 应用程序。请参阅使用 Helm 创建 JBoss EAP 7 应用程序。

流程

1. 在导航菜单中点 **Topology**。
   在 **Topology** 视图中，**eap74** 部署在右下角显示一个代码图标。

   此图标代表相关代码的 Git 存储库，或者安装了适当的运算符，它将在 IDE 中打开相关的代码。

2. 如果显示的图标是 CodeReady Workspaces 或 Eclipse Che，请点击它在 IDE 中启动关联的代码。否则，点它导航到关联的 Git 存储库。

验证

- 验证您可以在 Git 存储库或 IDE 中看到与应用程序关联的代码。

3.5. 查看构建状态

使用 Helm Chart 成功创建 JBoss EAP 7 应用程序后，您可以查看构建状态。

先决条件

- 已使用 Helm Chart 创建 JBoss EAP 7 应用程序。请参阅使用 Helm 创建 JBoss EAP 7 应用程序。

流程

1. 在导航菜单中点 **Topology**。
2. 在 **Topology** 视图中，点 **D eap74** 图标。
   此时会打开一个侧面板，其中包含有关应用程序的详细信息。
3. 在侧面面板中，点 **Resources** 选项卡。**Builds** 部分显示与应用程序构建相关的所有详细信息。

注意

JBoss EAP 7 应用程序以两个步骤构建：

- 第一个构建配置 **eap74-build-artifacts** 编译并打包 Jakarta EE 应用，并创建一个 JBoss EAP 服务器。应用在此 JBoss EAP 服务器上运行。

完成构建可能需要几分钟时间。构建通过各种状态（如 **Pending**、**Running** ）进行 **进度**。构建状态由相关消息表示。

构建完成后，会显示复选标记并显示以下消息： **Build the1 is complete**

- 第二个构建配置 **eap74** 将 Jakarta EE 部署和 JBoss EAP 服务器放在运行时镜像中，该镜像仅包含运行应用所需的内容。

第二个构建完成后，会显示复选标记并显示以下消息： **Buildbang2 has complete**

- 当第一个构建完成后，第二个构建将启动。

验证

- 验证 **eap74-build-artifacts** 和 **eap74** 的两个构建已完成：
  - 对于 **eap74-build-artifacts** 构建配置，会显示 **Build1141 is complete** 信息。
  - 对于 **eap 74 构建配置，会显示 Build** bang2 信息。

3.6. 查看 POD 状态

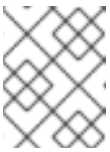
使用 Helm Chart 成功创建 JBoss EAP 7 应用程序后，您可以查看 pod 状态。

先决条件

- 已使用 Helm Chart 创建 JBoss EAP 7 应用程序。请参阅使用 Helm 创建 JBoss EAP 7 应用程序。

流程

1. 在导航菜单中点 **Topology**。
2. 在 **Topology** 视图中，点 **D eap74**。
   此时会打开一个侧面板，其中包含有关应用程序的详细信息。
3. 在 Details 选项卡中，将鼠标悬停在 pod 上，以在工具提示中查看 pod 状态。
   - pod 数量显示在 pod 圆圈中。
   - pod circle 的颜色表示 pod status: **Light blue = Pending**, **Blue = Not Ready**, **Dark blue = Running**。



注意

在 **Topology** 视图中，**D eap74** 部署图标的 dark outer circle 也表示 pod 状态。

验证

- 验证 pod circle 中的文本是否显示 **1 个 pod**。
- 当您将鼠标悬停在其中时，验证 Pod circle 是否显示 **1 Running**。

3.7. 运行 JBOSS EAP 7 应用

使用 Helm 成功创建并构建 JBoss EAP 7 应用程序后，您可以访问它。

先决条件

- 您已创建了 JBoss EAP 7 应用。请参阅使用 Helm 创建 JBoss EAP 7 应用程序 。

流程

- 在 **Topology 视图中**，单击右上角的外部链接图标打开 URL，并在一个单独的浏览器窗口中运行应用程序。

注意

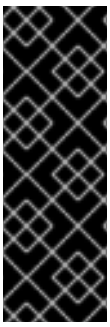
此操作会在 Web 浏览器窗口中打开 URL。

验证

- 验证 **Red Hat OpenShift 上的应用 JBoss EAP 7**是否在一个单独的浏览器窗口中打开。

第 4 章 为 JAVA 应用程序配置 JBOSS EAP

用于 OpenShift 的 JBoss EAP 映像已预配置为 Java 应用的基本用途。不过，您可以在镜像内配置 JBoss EAP 实例。推荐的方法是使用 OpenShift S2I 流程，以及应用模板参数和环境变量。



重要

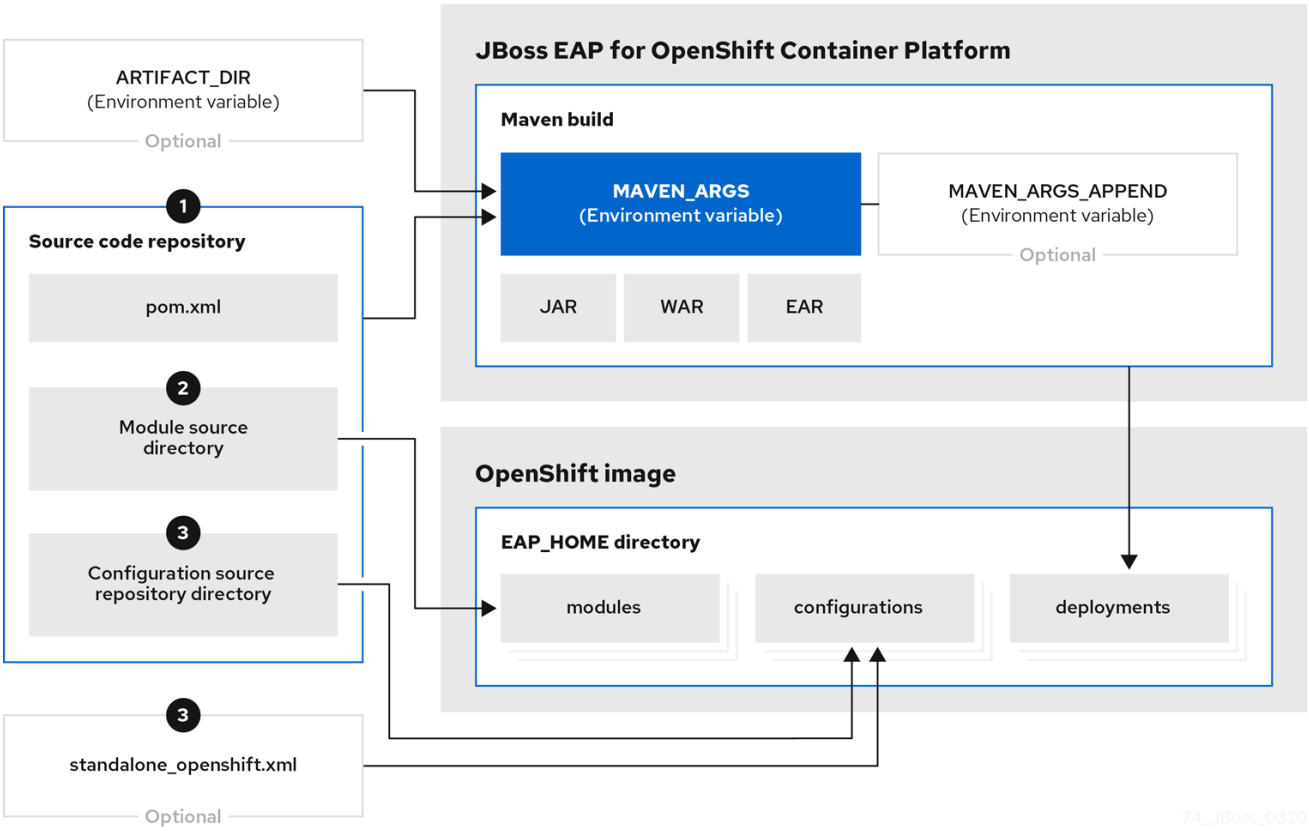
当容器重启或终止时，运行中的容器上所做的任何配置更改都将丢失。

这包括使用传统 JBoss EAP 安装中包含的脚本进行的任何配置更改，如 **add-user.sh** 或管理 CLI。

强烈建议您使用 OpenShift S2I 流程以及应用模板参数和环境变量，在 JBoss EAP for OpenShift 镜像内对 JBoss EAP 实例进行任何配置更改。

4.1. 用于 OPENSHIFT S2I 流程的 JBOSS EAP 如何工作

说明 JBoss EAP 的 S2I 进程的流程图：



1. 如果源代码存储库中存在 **pom.xml** 文件，S2I 构建器镜像将启动 Maven 构建流程。Maven 构建使用 **$MAVEN_ARGS** 的内容。
   如果源代码存储库中不存在 **pom.xml** 文件，S2I 构建器镜像将启动二进制类型构建。

   要添加自定义 Maven 参数或选项，请使用 **$MAVEN_ARGS_APPEND**。**$MAVEN_ARGS_APPEND** 变量将选项附加到 **$MAVEN_ARGS**。

   默认情况下，OpenShift 配置文件使用 Maven **软件包** 目标，其中包括用于跳过测试的系统属性(**-DskipTests**)和启用红帽 GA 存储库(**-Dcom.redhat.xpaas.repo**)。

Maven 构建成功的结果复制到 OpenShift 镜像 JBoss ***EAP 中的 EAP_HOME*/standalone/deployments/** 目录中。这包括 **$ARTIFACT_DIR** 环境变量指定的源存储库中的所有 JAR、WAR 和 EAR 文件。**ARTIFACT_DIR** 的默认值是 Maven 目标目录。

注意

要在 JBoss EAP 上的代理后面将 Maven 用于 OpenShift 镜像，请设置 **$HTTP_PROXY_HOST** 和 **$HTTP_PROXY_PORT** 环境变量。另外，您还可以设置 **$HTTP_PROXY_USERNAME**、**$HTTP_PROXY_PASSWORD** 和 **$HTTP_PROXY_NONPROXYHOSTS** 变量。

2. **模块** 源存储库目录中的所有文件复制到 JBoss ***EAP for OpenShift 镜像的 EAP_HOME*/modules/** 目录中。

3. **configuration** 源存储库目录中的所有文件复制到 JBoss ***EAP for OpenShift 镜像中的 EAP_HOME*/standalone/configuration/** 目录中。如果要使用自定义 JBoss EAP 配置文件，请将文件命名为 **standalone-openshift.xml**。

其它资源

- 如需有关二进制类型构建的更多信息，请参阅 OpenShift 4.2 文档上的 Binary（本地）源。
- 如需有关如何指示 S2I 进程使用自定义 Maven 工件存储库镜像的其他指导，请参阅 Artifact Repository Mirrors。

4.2. 使用环境变量为 OPENSHIFT 配置 JBOSS EAP

使用环境变量是为 OpenShift 镜像配置 JBoss EAP 的推荐方法。如需有关为应用容器和构建容器指定环境变量的说明，请参阅 OpenShift 文档。

例如，您可以在创建 OpenShift 应用程序时使用环境变量设置 JBoss EAP 实例的管理用户名和密码：

```
oc new-app --template=eap74-basic-s2i \
 -p IMAGE_STREAM_NAMESPACE=eap-demo \
 -p SOURCE_REPOSITORY_URL=https://github.com/jboss-developer/jboss-eap-quickstarts \
 -p SOURCE_REPOSITORY_REF=7.4.x \
 -p CONTEXT_DIR=kitchensink \
 -e ADMIN_USERNAME=myspecialuser \
 -e ADMIN_PASSWORD=myspecialp@ssw0rd
```

参考信息中列出了 JBoss EAP for OpenShift 镜像的可用环境变量。

4.2.1. JVM 内存配置

OpenShift EAP 镜像具有根据当前环境自动计算默认 JVM 内存设置的机制，但也可以使用环境变量配置 JVM 内存设置。

4.2.1.1. JVM 默认内存设置

如果为当前容器定义了内存限值，并且限制低于可用内存总量，则会自动计算默认 JVM 内存设置。否则，默认 JVM 内存设置是 EAP 版本的 **standalone.conf** 文件中定义的默认文件，用作镜像的基础服务器。

容器内存限值从文件 **/sys/fs/cgroup/memory/memory.limit_in_bytes** 中检索。使用 **/proc/meminfo** 命令检索可用内存总量。

当自动计算内存设置时，会使用以下公式：

- 最大堆大小(-Xmx)： 50% 的用户内存
- 初始堆大小(-Xms)：计算的最大堆大小的 2 5%(25%)

例如，定义的内存限值为 1 GB，这个限制低于 **/proc/meminfo** 报告的总可用内存，然后内存设置将为：-Xms128m -Xmx512

您可以使用以下环境变量来修改自动计算的 JVM 设置：请注意，这些变量仅在自动计算默认内存大小时才使用（换句话说，定义有效的容器内存限值）。

- **JAVA_MAX_MEM_RATIO**
- **JAVA_INITIAL_MEM_RATIO**
- **JAVA_MAX_INITIAL_MEM**

您可以通过将以下两个环境变量的值设置为 0 来禁用自动内存计算。

- **JAVA_INITIAL_MEM_RATIO**
- **JAVA_MAX_MEM_RATIO**

4.2.1.2. JVM Garbage Collection 设置

OpenShift 的 EAP 镜像包含垃圾回收和垃圾回收日志记录的设置

垃圾**收集器**设**置**

-XX:+UseParallelOldGC -XX:MinHeapFreeRatio=10 -XX:MaxHeapFreeRatio=20 -XX:GCTimeRatio=4 -XX:AdaptiveSizePolicyWeight=90 -XX:+ExitOnOutOfMemoryError

Java 8 的垃圾**收集日志**记录设**置（非模式 JVM）**

-verbose:gc -Xloggc:/opt/eap/standalone/log/gc.log -XX:+PrintGCDetails -XX:+PrintGCDateStamps -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=5 -XX:GCLogFileSize=3M -XX:-TraceClassUnloading

Java 11 的垃圾**收集日志**记录设**置（模**态 **JVM）**

-Xlog:gc*:file=/opt/eap/standalone/log/gc.log:time,uptimemillis:filecount=5,filesize=3M

4.2.1.3. **默**认设**置中的**资**源限制**

如果设置，镜像中会包含其他默认设置。

-XX:ParallelGCThreads={core-limit} -Djava.util.concurrent.ForkJoinPool.common.parallelism={core-limit} -XX:CICompilerCount=2

{core-limit} 的值使用 **JAVA_CORE_LIMIT** 环境变量或容器实施的 CPU 核心限制来定义。

CICompilerCount 的值始终固定为 *2*。

4.2.1.4. JVM 环**境**变**量**

使用这些环境变量，在 EAP 中为 OpenShift 镜像配置 JVM。

表 4.1. JVM 环境变量

变量名称	示例	默认值	JVM 设置	描述
JAVA_OPTS	-verbose:class	没有默认	多个	要传递给 **java** 命令的 JVM 选项。 使用 **JAVA_OPTS_APPEND** 配置额外的 JVM 设置。如果使用 **JAVA_OPTS**，一些不可配置的默认值不会添加到服务器 JVM 设置中。您必须明确添加这些设置。 使用 **JAVA_OPTS** 时，将禁用容器脚本默认添加的某些设置。禁用的设置包括 ● -XX:MetaspaceSize=96M ● -Djava.net.preferIPv4Stack=true ● -Djboss.modules.system.pkgs=jdk.nashorn.api,com.sun.crypto.provider ● -Djava.awt.headless=true 此外，如果没有启用自动内存计算，则不定义初始 Java 内存(-Xms)和最大 Java 内存(-Xmx)。

变量名称	示例	默认值	JVM 设置	描述
				如果您使用 **JAVA_OPTS** 配置附加设置，请添加这些默认值。
JAVA_OPTS_APPEND	-Dsome.property=value	没有默认	多个	用户指定的 Java 选项，可附加到 **JAVA_OPTS** 中生成的选项.
JAVA_MAX_MEM_RATIO	50	50	-Xmx	如果未在 **JAVA_OPTS** 中指定 **-Xmx** 选项，则使用此变量。此变量的值用于根据容器的限制计算默认的最大堆内存大小。如果在没有内存约束的容器中使用此变量，则 变量无效。如果在具有内存约束的容器中使用此变量，则 **-Xmx** 的值被设置为容器可用内存的指定比率。默认值为 *50*，表示可用内存的 50% 用作上限。要跳过最大内存计算，将此变量的值设置为 *0*。No **-Xmx** 选项将添加到 **JAVA_OPTS**。

变量名称	示例	默认值	JVM 设置	描述
JAVA_INITIAL_MEM_RATIO	25	25	-xms	如果未在 **JAVA_OPTS** 中指定 **-Xms** 选项，则使用此变量。此变量的值用于根据最大堆内存计算默认初始堆内存大小。如果在没有内存约束的容器中使用此变量，则 变量无效。如果在具有内存约束的容器中使用此变量，则 **-Xms** 的值被设置为 **-Xmx** 内存的指定比率。默认值为 *25*，表示最大内存的 25% 用作初始堆大小。要跳过初始内存的计算，将此变量的值设置为 *0*。No **-Xms** 选项将添加到 **JAVA_OPTS**。
JAVA_MAX_INITIAL_MEM	4096	4096	-xms	如果未在 **JAVA_OPTS** 中指定 **-Xms** 选项，则使用此变量。此变量的值用于计算初始内存堆的最大大小。该值以兆字节(MB)为单位表示。如果在没有内存约束的容器中使用此变量，则 变量无效。如果此变量在具有内存约束的容器中使用，则 **-Xms** 的值将设置为 变量中指定的值。默认值为 4096MB，指定最大初始堆永远不会大于 4096MB。

变量名称	示例	默认值	JVM 设置	描述
JAVA_DIAGNOSTICS	true	false (disabled)	设置取决于容器使用的 JDK。 ● OpenJDK 8: -XX:NativeMemoryTracking=summary -XX:+PrintGC -XX:+PrintGCDateStamps -XX:+PrintGCTimeStamps -XX:+UnlockDiagnosticVMOptions ● OpenJDK 11: -Xlog:gc:utctime -XX:NativeMemoryTracking=summary	将此变量的值设为 *true*，以在发生事件时将诊断信息包含在标准输出中。如果在 **JAVA_DIAGNOSTICS** 已定义为 *true* 的环境中将此变量定义为 *true*，则仍会包含诊断。
DEBUG	true	false	-agentlib:jdwp=transport=dt_socket,address=$DEBUG_PORT,server=y,suspend=n	启用远程调试。
DEBUG_PORT	8787	8787	-agentlib:jdwp=transport=dt_socket,address=$DEBUG_PORT,server=y,suspend=n	指定用于调试的端口。

变量名称	示例	默认值	JVM 设置	描述
JAVA_CORE_LIMIT		未定义	-XX:parallelGCThreads -Djava.util.concurrent.ForkJoinPool.common.parallelism -XX:CICompilerCount	内核数的用户定义的限制。如果容器报告了限制约束，则 JVM 设置的值仅限于容器核心限值。-XXCICompilerCount 的值始终为 *2*。默认情况下，此变量未定义。在这种情况下，如果容器中没有定义限制，则不会设置 JVM 设置。
GC_MIN_HEAP_FREE_RATIO	20	10	-XX:MinHeapFreeRatio	垃圾回收后最少空闲堆的百分比以避免扩展。
GC_MAX_HEAP_FREE_RATIO	40	20	-XX:MaxHeapFreeRatio	垃圾回收后的最大堆百分比以避免收缩。
GC_TIME_RATIO	4	4	-XX:GCTimeRatio	指定垃圾回收之外所用时间（例如，应用程序执行时间）与垃圾回收所花费的时间比例。
GC_ADAPTIVE_SIZE_POLICY_WEIGHT	90	90	-XX:AdaptiveSizePolicyWeight	给予当前垃圾收集时间与之前的垃圾回收时间的权重。
GC_METASPACE_SIZE	20	96	-XX:MetaspaceSize	初始元空间大小。
GC_MAX_METASPACE_SIZE	100	256	-XX:MaxMetaspaceSize	最大元空间大小。
GC_CONTAINER_OPTIONS	-XX:+UseG1GC	-XX:-UseParallelOldGC	-XX:-UseParallelOldGC	指定要使用的 Java 垃圾回收。变量的值应当是 JRE 命令行选项，用于指定所需的垃圾回收。指定的 JRE 命令覆盖默认值。

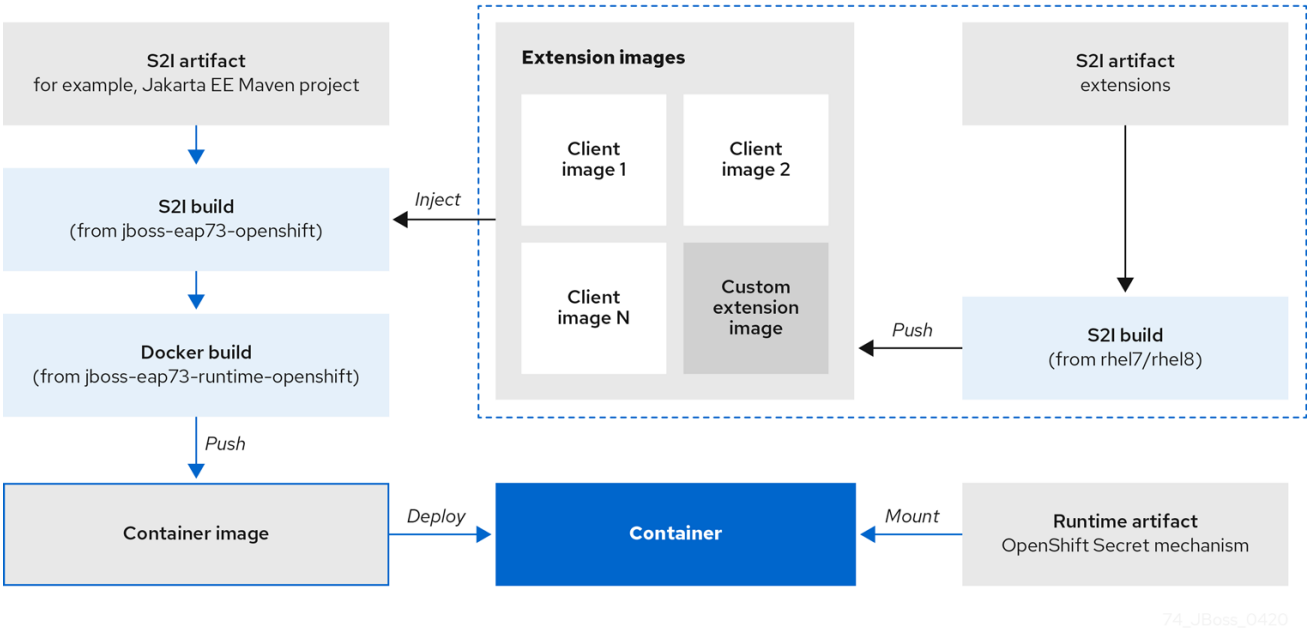
以下环境变量已弃用：

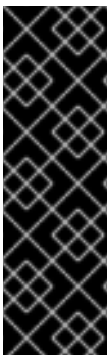
- **JAVA_OPTIONS** ：使用 **JAVA_OPTS**。
- **INITIAL_HEAP_PERCENT** ：使用 **JAVA_INITIAL_MEM_RATIO**.
- **container_HEAP_PERCENT** ：使用 **JAVA_MAX_MEM_RATIO**。

4.3. 构建扩展和项目工件

OpenShift 镜像的 JBoss EAP 使用各种构件扩展 OpenShift 中的数据库支持。这些工件通过不同的机制包含在构建的镜像中：

- S2I 工件，这些工件在 S2I 过程中注入到镜像中。
- 通过 OpenShift 机密机制提供的环境文件的运行时构件。





重要

现在，弃用了对将红帽提供的内部数据源驱动程序与 JBoss EAP 搭配使用的支持。红帽建议您将数据库供应商获取的 JDBC 驱动程序用于 JBoss EAP 应用。

JBoss EAP for OpenShift 镜像不再提供以下内部数据源：

- MySQL
- PostgreSQL

有关安装驱动程序的更多信息，请参阅模块、驱动程序和通用部署。

有关使用 JBoss EAP 配置 JDBC 驱动程序的更多信息，请参阅 JBoss EAP *配置指南中的* JDBC 驱动程序。

请注意，如果您想将其添加到置备的服务器中，您也可以创建自定义层来安装这些驱动程序和数据源。

其它资源

针对 OpenShift 的 JBoss EAP 的功能调试

4.3.1. S2I Artifacts

S2I 工件包括模块、驱动程序和其他通用部署，它们提供了部署所需的必要配置基础架构。此配置在 S2I 过程中构建至镜像中，因此在运行时仅需要配置数据源和相关资源适配器。

如需有关如何指示 S2I 流程利用自定义 Maven 工件存储库镜像的其他指导，请参阅 Artifact Repository Mirrors。

4.3.1.1. 模块、驱动程序和通用部署

对于 OpenShift 镜像，在 JBoss EAP 中包括这些 S2I 工件的几个选项：

1. 将工件包含在应用源部署目录中。构建期间下载工件并注入到镜像中。这类似于在用于 OpenShift 镜像的 JBoss EAP 上部署应用。
2. 包含 **CUSTOM_INSTALL_DIRECTORIES** 环境变量，这是在 S2I 过程中用于安装和配置镜像工件的目录列表。在 S2I 中包括此信息有两种方法：
   - 指定的 **安装目录中的 install.sh** 脚本。安装脚本在 S2I 过程中执行，并以强制方式运行。

     install.sh 脚本示例

     ```
     #!/bin/bash

     injected_dir=$1
     source /usr/local/s2i/install-common.sh
     install_deployments ${injected_dir}/injected-deployments.war
     install_modules ${injected_dir}/modules
     configure_drivers ${injected_dir}/drivers.env
     ```

     install.sh 脚本负责使用 **install-common.sh** 提供的 API 来自定义基础镜像。**install-common.sh** 包含 **install.sh** 脚本用于安装和配置模块、驱动程序和通用部署的功能。

     install-common.sh 中包含的功能：

     - **install_modules**
     - **configure_drivers**
     - **install_deployments**

       模块

       模块是用于类加载和依赖关系管理的逻辑类分组。模块在应用服务器的 ***EAP_HOME*/modules/** 目录中定义。每个模块都作为子目录存在，如 ***EAP_HOME*/modules/org/**。每个模块目录随后包含一个插槽子目录，默认为 main，它包含 **module.xml** 配置文件以及任何所需的 JAR 文件。

       有关为 MySQL 和 PostgreSQL JDBC 驱动程序配置 **module.xml** 文件的更多信息，请参阅 JBoss EAP 配置指南中的数据源配置示例。

       PostgreSQL Datasource 的 module.xml 文件示例

       ```
       <?xml version="1.0" encoding="UTF-8"?>
       <module xmlns="urn:jboss:module:1.0" name="org.postgresql">
       <resources>
       ```

```
<resource-root path="postgresql-jdbc.jar"/>
</resources>
<dependencies>
<module name="javax.api"/>
<module name="javax.transaction.api"/>
</dependencies>
</module>
```

MySQL Connect/J 8 数据源的 module.xml 文件示例

```
<?xml version="1.0" encoding="UTF-8"?>
<module xmlns="urn:jboss:module:1.0" name="com.mysql">
<resources>
<resource-root path="mysql-connector-java-8.0.Z.jar" />
</resources>
<dependencies>
<module name="javax.api"/>
<module name="javax.transaction.api"/>
</dependencies>
</module>
```

注意

mysql-connector-java-8.0.Z.jar 中的 ".Z " 指示下载的 **JAR** 文件的版本。可以重命名文件，但名称必须与 **module.xml** 文件中的名称匹配。

install .sh 中的 install _modules 功能将对应的 JAR 文件复制到 JBoss EAP 中的模块目录中，以及 **module.xml**。

驱动**程序**

驱动程序作为模块安装。该驱动程序随后由 **configure_drivers** 功能在 **install.sh** 中配置，其配置属性 在运行时工件 环境文件中定义。

添加数据源驱动**程序**

MySQL 和 PostgreSQL 数据源不再作为预配置的内部数据源提供。您仍可将这些驱动程序安装为模块；请参阅模块、驱动程序和通用部署中的描述。您可以从数据库供应商为 JBoss EAP 应用获取这些 JDBC 驱动程序。

为 **要安装的每个数据源创建一个 driver.env** 文件。

MySQL Datasource 的 Driver.env 文件示例

```
#DRIVER
DRIVERS=MYSQL
MYSQL_DRIVER_NAME=mysql
MYSQL_DRIVER_MODULE=org.mysql
MYSQL_DRIVER_CLASS=com.mysql.cj.jdbc.Driver
MYSQL_XA_DATASOURCE_CLASS=com.mysql.cj.jdbc.MysqlXADataSource
```

PostgreSQL Datasource 的 driver.env 文件示例

```
#DRIVER
```

```
DRIVERS=POSTGRES
POSTGRES_DRIVER_NAME=postgresql
POSTGRES_DRIVER_MODULE=org.postgresql
POSTGRES_DRIVER_CLASS=org.postgresql.Driver
POSTGRES_XA_DATASOURCE_CLASS=org.postgresql.xa.PGXADataSource
```

有关下载各种驱动程序的位置（如 MySQL 或 PostgreSQL）的详情，请参阅《配置指南》中的 JDBC 驱动程序下载位置。

通用部署

可以使用 install**-common.sh** 中的 API 提供的 **install_deployments** 功能，从注入的镜像部署 JAR、WAR、RAR 或 EAR 等可部署存档文件。

- 如果已声明 **CUSTOM_INSTALL_DIRECTORIES** 环境变量，但没有在自定义安装目录中找到 **install.sh** 脚本，则以下工件目录将复制到构建镜像中的对应目的地：
  - **modules/*** 复制到 **$JBOSS_HOME/modules/**
  - **configuration/*** 复制到 **$JBOSS_HOME/standalone/configuration**
  - **Deployment/*** 复制到 **$JBOSS_HOME/standalone/deployments**

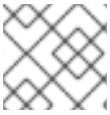
  与 **install.sh** 替代方案相比，这是一种基本的配置方法，并且要求构件进行适当的结构。

4.3.2. runtime Artifacts

4.3.2.1. datasources

数据源有两种类型：

1. 内部数据源.这些数据源在 OpenShift 上运行，但默认情况下无法通过 Red Hat Registry 或 OpenShift 存储库中提供。这些数据源的配置由添加到 OpenShift Secret 的环境文件提供。
2. 外部数据源.这些数据源不在 OpenShift 上运行。外部数据源的配置由添加到 OpenShift Secret 的环境文件提供。



注意

有关创建和配置 OpenShift Secret 的更多信息，请参阅 Secret。

您可以在目录中创建数据源环境文件，如源项目的配置目录。以下示例显示了数据源环境文件的内容：

示例：数据源环境文件

```
DB_SERVICE_PREFIX_MAPPING=PostgresXA-POSTGRES=DS1
DS1_JNDI=java:jboss/datasources/pgds
DS1_DRIVER=postgresql-42.2.5.jar
DS1_USERNAME=postgres
DS1_PASSWORD=postgres
DS1_MAX_POOL_SIZE=20
DS1_MIN_POOL_SIZE=20
DS1_CONNECTION_CHECKER=org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLValidCo
```

```
nnectionChecker
DS1_EXCEPTION_SORTER=org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLExceptionSo
rter
```

DB_SERVICE_PREFIX_MAPPING 属性是一个以逗号分隔的数据源属性前缀列表。然后，这些前缀附加到该数据源的所有属性中。然后，可以将多个数据源包含在单个环境文件中。另外，每个数据源也可以在单独的环境文件中提供。

数据源包含两种类型的属性：连接池特定属性和数据库驱动程序特定属性。连接池相关的属性生成与数据源的连接。数据库驱动程序特定属性决定数据源的驱动程序，并配置为驱动程序 S2I 工件。

在上例中，**DS1** 是数据源前缀，**CONNECTION_CHECKER** 指定一个用于验证数据库连接的连接检查器类，**EXCEPTION_SORTER** 指定用于检测严重数据库连接异常的分类器类。

数据源环境文件添加到项目的 OpenShift Secret 中。然后，模板中使用 **ENV_FILES environment** 属性调用这些环境文件，其值是完全限定环境文件的逗号分隔列表，如下所示：

```
{
    “Name”: “ENV_FILES”,
    “Value”: “/etc/extensions/datasources1.env,/etc/extensions/datasources2.env”
}
```

4.3.2.2. 资源适配器

资源适配器配置由添加到 OpenShift Secret 的环境文件提供。

表 4.2. 资源适配器属性

属性	描述
*PREFIX*_ID	服务器配置文件中指定的资源适配器标识符。
*前缀*_ARCHIVE	资源适配器存档。
*PREFIX*_MODULE_SLOT	插槽子目录，其中包含 **module.xml** 配置文件和任何所需的 JAR 文件。
*PREFIX*_MODULE_ID	JBoss 模块 ID，可以从中加载对象工厂 Java 类。
*前缀*_*CONNECTION_CLASS*	受管连接工厂或 admin 对象的完全限定类名称。
*PREFIX*_CONNECTION_JNDI	连接工厂的 JNDI 名称。
*PREFIX*_PROPERTY_ParentDirectory	存储数据文件的目录。
*PREFIX*_PROPERTY_AllowParentPaths	将 **AllowParentPaths** 设置为 **false** 以禁止。 在路径中。这可以防止请求父目录中不包含的文件。
*PREFIX*_POOL_MAX_SIZE	池的最大连接数。不会在每个子池中创建更多连接。
*PREFIX*_POOL_MIN_SIZE	池的最小连接数。

属性	描述
*PREFIX*_POOL_PREFILL	指定是否应预先填充池。更改此值需要重新启动服务器。
前缀_POOL_FLUSH_STRATEGY	出现错误时，应如何清空池。有效值为：Fa **ilingConnectionOnly**（默认）、**IdleConnections** 和 **EntirePool**。

RESOURCE_ADAPTERS 属性是一个以逗号分隔的资源适配器属性前缀列表。然后，这些前缀会附加到该资源适配器的所有属性中。然后，可将多个资源适配器包含在单个环境文件中。在以下示例中，**MYRA** 用作资源适配器的前缀。另外，每个资源适配器也可以在单独的环境文件中提供。

示例：资源适配器环境文件

```
#RESOURCE_ADAPTER
RESOURCE_ADAPTERS=MYRA
MYRA_ID=myra
MYRA_ARCHIVE=myra.rar
MYRA_CONNECTION_CLASS=org.javaee7.jca.connector.simple.connector.outbound.MyManagedCo
nnectionFactory
MYRA_CONNECTION_JNDI=java:/eis/MySimpleMFC
```

资源适配器环境文件添加到项目命名空间的 OpenShift Secret 中。然后，模板中使用 **ENV_FILES environment** 属性调用这些环境文件，其值是完全限定环境文件的逗号分隔列表，如下所示：

```
{
    "Name": "ENV_FILES",
    "Value": "/etc/extensions/resourceadapter1.env,/etc/extensions/resourceadapter2.env"
}
```

4.4. 将 JBOSS EAP 模板用于 OPENSHIFT 的结果

当您使用 JBoss EAP 模板编译应用时，可能会生成两个镜像。

在创建最终镜像 **[application name]-build-artifacts** 之前，可能会生成名为 **[application name]** 的中间镜像。

您可以在部署应用程序后删除 **[application name]-build-artifacts** 镜像。

4.5. 用于 OPENSHIFT 镜像的 RED HAT JBOSS ENTERPRISE APPLICATION PLATFORM 的 SSO 配置

在适用于 OpenShift 镜像的红帽 JBoss 企业应用平台中，SSO 配置为使用传统 **安全** 子系统。

这些镜像中将 environmment 变量 **SSO_FORCE_LEGACY_SECURITY** 设置为 **true**。

如果要 **将 elytron** 子系统用于 SSO 安全，请将 **SSO_FORCE_LEGACY_SECURITY** 环境变量的值更新为 **false**。

4.6. 默认数据源

JBoss EAP 7.4 中不提供数据源 **示例DS**。

有些快速入门需要这个数据源：

- **cmt**
- **thread-racing**

由客户开发的应用可能还需要 **ExampleDS** 数据源。

如果您需要默认数据源，请使用 **GENERATE_DEFAULT_DATASOURCE** 环境变量在调配 JBoss EAP 服务器时包括它。

```
ENABLE_GENERATE_DEFAULT_DATASOURCE=true
```

第 5 章 OPENSHIFT 的 JBOSS EAP 的功能

构建包含 JBoss EAP 的镜像时，您可以控制要包含在镜像中的 JBoss EAP 功能和子系统。

S2I 镜像中包含的默认 JBoss EAP 服务器包括完整的服务器和所有功能。您可能想要减少调配的服务器中包含的功能。例如，您可能要减少调配的服务器的安全暴露，或者您可能要减小内存占用空间，使其更适合微服务容器。

5.1. 配置自定义 JBOSS EAP 服务器

要调配具有修剪功能的自定义服务器，请在 S2I 构建阶段传递 **GALLEON_PROVISION_LAYERS** 环境变量。

环境变量的值是用于置备以构建服务器的层的逗号分隔列表。

例如，如果您将环境变量指定为 **GALLEON_PROVISION_LAYERS=jaxrs-server,sso**，则会为 JBoss EAP 服务器置备以下功能：

- Servlet 容器
- 配置数据源的功能
- **jaxrs**、**we** ld 和 **jpa** 子系统
- Red Hat SSO 集成

5.2. 可用的 JBOSS EAP 层

红帽提供六个层来自定义 OpenShift 中的 JBoss EAP 服务器调配。

三个层是提供核心功能的基础层。有三个是增强基础层的 decorator 层。

任何调配层不支持以下 Jakarta EE 规格：

- Jakarta Server Faces 2.3
- Jakarta Enterprise Beans 3.2
- Jakarta XML Web Services 2.3

5.2.1. 基本层

每个基础层都包含适用于典型服务器用户案例的核心功能。

datasources-web-server

此层包括一个 servlet 容器，以及配置数据源的功能。

以下是 **datasources-web-server** 中默认包含的 JBoss EAP 子系统：

- **core-management**
- **datasources**
- **deployment-scanner**

- **EE**
- **elytron**
- **io**
- **jca**
- **jmx**
- **logging**
- **naming**
- **request-controller**
- **security-manager**
- **事务**
- **undertow**

此层支持以下 Jakarta EE 规格：

- Jakarta JSON Processing 1.1
- Jakarta JSON Binding 1.0
- Jakarta Servlet 4.0
- Jakarta Expression Language 3.0
- Jakarta 服务器页面 2.3
- Jakarta Standard Tag Library 1.2
- jakarta Concurrency 1.1
- Jakarta Annotations 1.3
- Jakarta XML Binding 2.3
- 雅加达调试支持其他语言 1.0
- Jakarta Transactions 1.3
- Jakarta Connectors 1.7

jaxrs-server

该层通过以下 JBoss EAP 子系统 **增强了数据源-web-server** 层：

- **jaxrs**
- **weld**
- **jpa**

此层还添加了基于 Infinispan 的第二级实体在容器中进行本地缓存。

除了 **datasources-web-server** 层所支持的以下 Jakarta EE 规格外，还支持以下 Jakarta EE 规格：

- Jakarta 上下文和依赖注入 2.0
- Jakarta Bean 验证 2.0
- Jakarta Interceptors 1.2
- Jakarta RESTful Web Services 2.1
- Jakarta Persistence 2.2

cloud-server

该层使用以下 JBoss EAP 子系统增强了 **jaxrs-server** 层：

- **resource-adapters**
- **messaging-activemq** （远程代理消息传递，而非嵌入式消息传递）

此层还会在 **jaxrs-server** 层中添加以下可观察功能：

- 健康子系统
- 指标子系统

除了 **jaxrs-server** 层支持的以下 Jakarta EE 规格外，还支持以下 Jakarta EE 规格：

- Jakarta 安全 1.0

5.2.2. decorator Layers

解码器层不单独使用。您可以使用基础层配置一个或多个 decorator 层，以提供额外的功能。

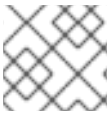
sso

这个解码器层将红帽单点登录集成添加到调配的服务器中。

Observability（可观察性）

这个 decorator 层在置备的服务器中添加以下可观察功能：

- 健康子系统
- 指标子系统



注意

此层内置到 **cloud-server** 层。您不需要将此层添加到 **云服务器层**。

web-clustering

此层将嵌入式 Infinispan 型 Web 会话集群添加到调配的服务器。

5.3. 在 JBOSS EAP 中置备用户开发层

除了红帽提供的调配层外，您还可以配置您开发的自定义层。

流程

1. 使用 Galleon Maven 插件构建自定义层。
   如需更多信息，请参阅准备 Maven 项目。

2. 将自定义层部署到可访问的 Maven 存储库。

3. 您可以使用自定义 Galleon 功能包环境变量在 S2I 镜像构建过程中自定义 Galleon 功能包和层。
   有关自定义 Galleon 功能包和层的更多信息，请参阅在 S2I 构建期间使用自定义 Galleon 功能包。

4. *可选：* 创建一个自定义调配文件来引用用户定义的层并支持的 JBoss EAP 层并将其存储在应用程序目录中。
   有关创建自定义配置文件的更多信息，请参阅为 JBoss EAP 自定义调配文件。

5. 运行 S2I 流程以在 OpenShift 中调配 JBoss EAP 服务器。
   如需更多信息，请参阅在 S2I 构建过程中使用自定义 Galleon 功能包。

5.3.1. 为 JBoss EAP 构建和部署 Galleon 层

自定义 Galleon 层打包在 Galleon 功能包中，该包设计为使用 JBoss EAP 7.4 运行。

在 Openshift 中，您可以构建并使用 Galleon 功能包，其中包含要置备的层，例如，JBoss EAP 7.4 服务器的 MariaDB 驱动程序和数据源。层包含服务器上安装的内容。层可以更新服务器 XML 配置文件，并在服务器安装中添加内容。

本节记录了如何在 OpenShift 中构建和使用 Galleon 功能包，其中包含层为 JBoss EAP 7.4 服务器置备 MariaDB 驱动程序和数据源。

5.3.1.1. 准备 Maven 项目

Galleon 功能包使用 Maven 创建。此流程包括创建新 Maven 项目的步骤。

流程

1. 要创建新 Maven 项目，请运行以下命令：

```
mvn archetype:generate -DarchetypeGroupId=org.codehaus.mojo.archetypes -
DarchetypeArtifactId=pom-root -DgroupId=org.example.mariadb -DartifactId=mariadb-
galleon-pack -DinteractiveMode=false
```

2. 在 **mariadb-galleon-pack** 目录中，更新 **pom.xml** 文件，使其包含 Red Hat Maven 存储库：

```
<repositories>
  <repository>
    <id>redhat-ga</id>
    <name>Redhat GA</name>
    <url>https://maven.repository.redhat.com/ga/</url>
  </repository>
</repositories>
```

3. 更新 **pom.xml** 文件，以添加 EAP Galleon 功能包和 MariaDB 驱动程序的依赖项：

```
<dependencies>
  <dependency>
    <groupId>org.jboss.eap</groupId>
```

```
    <artifactId>wildfly-ee-galleon-pack</artifactId>
    <version>7.4.4.GA-redhat-00011</version>
    <type>zip</type>
  </dependency>
  <dependency>
    <groupId>org.mariadb.jdbc</groupId>
    <artifactId>mariadb-java-client</artifactId>
    <version>3.0.5</version>
  </dependency>
</dependencies>
```

4. 更新 **pom.xml** 文件，使其包含用于构建 Galleon feature-pack 的 Maven 插件：

```
<build>
  <plugins>
    <plugin>
      <groupId>org.wildfly.galleon-plugins</groupId>
      <artifactId>wildfly-galleon-maven-plugin</artifactId>
      <version>5.2.11.Final</version>
      <executions>
        <execution>
          <id>mariadb-galleon-pack-build</id>
          <goals>
            <goal>build-user-feature-pack</goal>
          </goals>
          <phase>compile</phase>
        </execution>
      </executions>
    </plugin>
  </plugins>
</build>
```

5.3.1.2. 添加功能包内容

这个过程帮助您将层添加到自定义 Galleon 功能包中，例如：功能包，包括 MariaDB 驱动程序和数据源层。

先决条件

- 您已创建了一个 Maven 项目。如需了解更多详细信息，请参阅准备 Maven 项目。

流程

1. 在自定义功能包 Maven 项目中创建目录 **src/main/resources**，例如 准备 Maven 项目。该目录是包含 feature-pack 内容的根目录。
2. 创建 **src/main/resources/modules/org/mariadb/jdbc/main** 目录。
3. 在 **主目录中**，创建一个名为 **module.xml** 的文件，其内容如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<module name="org.mariadb.jdbc" xmlns="urn:jboss:module:1.8">
  <resources>
    <artifact name="${org.mariadb.jdbc:mariadb-java-client}"/> (1)
  </resources>
```

```
  <dependencies> (2)
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

(1) MariaDB 驱动程序 **groupId** 和 **artifactId**。在调配时，将安装实际的驱动程序 jar 文件。驱动程序的版本从 **pom.xml** 文件引用。

(2) MariaDB 驱动程序的 **JBoss** 模块模块依赖项。

4. 创建 **src/main/resources/layers/standalone/** 目录。这是 Galleon 功能包定义的所有层的根目录。

5. 创建 **src/main/resources/layers/standalone/mariadb-driver** 目录。

6. 在 **mariadb-driver** 目录中，使用以下内容创建 **layer-spec.xml** 文件：

```
<?xml version="1.0" ?>
<layer-spec xmlns="urn:jboss:galleon:layer-spec:1.0" name="mariadb-driver">
 <feature spec="subsystem.datasources"> (1)
      <feature spec="subsystem.datasources.jdbc-driver">
           <param name="driver-name" value="mariadb"/>
           <param name="jdbc-driver" value="mariadb"/>
           <param name="driver-xa-datasource-class-name"
value="org.mariadb.jdbc.MariaDbDataSource"/>
           <param name="driver-module-name" value="org.mariadb.jdbc"/>
      </feature>
 </feature>
 <packages> (2)
      <package name="org.mariadb.jdbc"/>
 </packages>
</layer-spec>
```

(1) 使用名为 MariaDB 的 JDBC-driver 更新 datasources 子sytem 配置，由模块 **org.mariadb.jdbc** 实施。

(2) 包含调配层时安装的驱动程序类的 **JBoss** 模块模块。

mariadb-driver 层使用由 **JBoss 模块模块** 实施的 JDBC 驱动程序配置更新数据源子系统。

7. 创建 **src/main/resources/layers/standalone/mariadb-datasource** 目录。

8. 在 **mariadb-datasource** 目录中，使用以下内容创建 **layer-spec.xml** 文件：

```
<?xml version="1.0" ?>
<layer-spec xmlns="urn:jboss:galleon:layer-spec:1.0" name="mariadb-datasource">
<dependencies>
    <layer name="mariadb-driver"/> (1)
</dependencies>

<feature spec="subsystem.datasources.data-source"> (2)
      <param name="data-source" value="MariaDBDS"/>
      <param name="jndi-name"
```

```
value="java:jboss/datasources/${env.MARIADB_DATASOURCE:MariaDBDS}"/>
    <param name="connection-url"
value="jdbc:mariadb://${env.MARIADB_HOST:localhost}:${env.MARIADB_PORT:3306}/${env.
MARIADB_DATABASE}"/> 3
    <param name="driver-name" value="mariadb"/>
    <param name="user-name" value="${env.MARIADB_USER}"/> 4
    <param name="password" value="${env.MARIADB_PASSWORD}"/>
</feature>
</layer-spec>
```

1. 此依赖项在调配数据源时强制实施 MariaDB 驱动程序的调配。在调配该层时，层依赖的所有层都会被自动置备。
2. 使用名为 MariaDBDS 的数据源更新数据源子系统配置。
3. 数据源的名称、主机、端口和数据库值从环境变量 ***MARIADB_DATASOURCE***、***MARIADB_HOST***、***MARIADB_PORT*** 和 ***MARIADB_DATABASE*** 解决，在服务器启动时设置。
4. 用户名和密码值从 ***MARIADB_USER*** 和 ***MARIADB_PASSWORD*** 中解析。

9. 运行以下命令构建 Galleon 功能包：

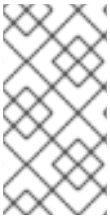
```
mvn clean install
```

创建文件 **target/mariadb-galleon-pack-1.0-SNAPSHOT.zip**。

5.3.1.3. 在 S2I 构建过程中使用自定义 Galleon 功能包

自定义功能包必须提供给 OpenShift S2I 构建期间发生的 Maven 构建。这通常通过将自定义功能包部署为工件来实现，例如 **org.example.mariadb:mariadb-galleon-pack:1.0-SNAPSHOT** 到可访问的 Maven 存储库。

要在部署前测试 feature-pack，您可以使用 EAP S2I 构建器镜像功能，允许您使用本地构建的 Galleon 功能包。使用以下步骤使用 MariaDB 驱动程序来自定义 **todo-backend** EAP Quickstart，而不是 PostgreSQL 驱动程序。



注意

- 有关 **todo-backend** EAP Quickstart 的更多信息，请参阅 EAP Quickstart。
- 有关为自定义 Galleon 功能包使用配置 JBoss EAP S2I 镜像的更多信息，请参阅使用 高级环境变量配置 Galleon。

先决条件

- 已安装 OpenShift 命令行
- 已登陆到 OpenShift 集群
- 已在集群中安装了 JBoss EAP OpenShift 镜像
- 您已配置了对 **Red Hat Container** registry 的访问。如需更多信息，请参阅 Red Hat Container Registry。

- 您已创建了自定义 Galleon 功能包。如需更多信息，请参阅准备 Maven 项目。

流程

1. 运行以下命令启动 MariaDB 数据库：

```
oc new-app -e MYSQL_USER=admin -e MYSQL_PASSWORD=admin -e
MYSQL_DATABASE=mariadb registry.redhat.io/rhscl/mariadb-101-rhel7
```

OpenShift service **mariadb-rhcsrhel7** 已创建并启动。

2. 在 Maven 项目目录 **mariadb-galleon-pack** 中运行以下命令来从 feature-pack ZIP 归档创建 secret，由自定义 feature-pack Maven 构建生成：

```
oc create secret generic mariadb-galleon-pack --from-file=target/mariadb-galleon-pack-1.0-
SNAPSHOT.zip
```

创建 secret **mariadb-galleon-pack**。启动 S2I 构建时，此机密用于在 pod 中挂载 feature-pack zip 文件，以便在服务器置备阶段提供该文件。

3. 要创建新的 OpenShift 构建来构建包含通过 Galleon 修剪的服务器内的 **todo-backend** quickstart 部署的应用程序镜像，请运行以下命令：

```
oc new-build jboss-eap74-openjdk11-openshift:latest~https://github.com/jboss-
developer/jboss-eap-quickstarts#EAP_7.4.0.GA \
--context-dir=todo-backend \
--env=GALLEON_PROVISION_FEATURE_PACKS="org.example.mariadb:mariadb-galleon-
pack:1.0-SNAPSHOT" \ 1
--env=GALLEON_PROVISION_LAYERS="jaxrs-server,mariadb-datasource" \ 2
--env=GALLEON_CUSTOM_FEATURE_PACKS_MAVEN_REPO="/tmp/repo" \ 3
--env=MAVEN_ARGS_APPEND="-Dcom.redhat.xpaas.repo.jbossorg" \
--build-secret=mariadb-galleon-pack:/tmp/repo/org/example/mariadb/mariadb-galleon-
pack/1.0-SNAPSHOT \ 4
--name=todos-app-build
```

1. 包含以逗号分隔的功能包 Maven 协调列表的自定义 feature-pack 环境变量，如 **groupId:artifactId:version**。
2. 用于置备服务器的 Galleon 层集合。**jaxrs-server** 是一个基本的服务器层，**mariadb-datasource** 是自定义层，它会将 MariaDB 驱动程序和新的数据源引入到服务器安装。
3. 包含 MariaDB 功能包的镜像中本地 Maven 存储库的位置。在镜像中挂载 secret 时，此软件仓库会被填充。
4. **mariadb-galleon-pack** secret 挂载到 **/tmp/repo/org/example/mariadb/mariadb-galleon-pack/1.0-SNAPSHOT** 目录中。

4. 要从创建的 OpenShift 构建启动新构建，请运行以下命令：

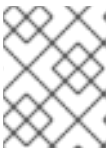
```
oc start-build todos-app-build
```

成功执行命令后，会创建镜像 **todos-app-build**。

5. 要创建新部署，请执行以下命令，提供将数据源绑定到正在运行的 MariaDB 数据库所需的环境变量：

```
oc new-app --name=todos-app todos-app-build \
--env=MARIADB_PORT=3306 \
--env=MARIADB_USER=admin \
--env=MARIADB_PASSWORD=admin \
--env=MARIADB_HOST=mariadb-101-rhel7 \
--env=MARIADB_DATABASE=mariadb  \
--env=MARIADB_DATASOURCE=ToDos (1)
```

(1) Quickstart 期望数据源命名为 **ToDos**



注意

有关自定义 Galleon 功能包环境变量的详情，请参阅自定义 Galleon 功能包环境变量

6. 要公开 **todos-app** 应用程序，请运行以下命令：

```
oc expose svc/todos-app
```

7. 要创建新任务，请运行以下命令：

```
curl -X POST http://$(oc get route todos-app --template='{{ .spec.host }}') \
  -H 'Content-Type: application/json' \
  -d '{"title":"todo1"}'
```

8. 要访问任务列表，请运行以下命令：

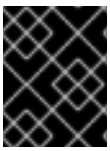
```
curl http://$(oc get route todos-app --template='{{ .spec.host }}')
```

添加的任务显示在浏览器中。

5.3.1.4. 用于 JBoss EAP 的自定义调配文件

自定义调配文件是文件名 **provisioning.xml** 的 XML 文件，存储在 **galleon** 子目录中。

使用 **provisioning.xml** 文件是 **GALLEON_PROVISION_FEATURE_PACKS** 和 **GALLEON_PROVISION_LAYERS** 环境变量的替代选择。在 S2I 构建期间， **provisioning.xml** 文件用于调配自定义 EAP 服务器。



重要

在使用 **GALLEON_PROVISION_LAYERS** 环境变量时，不要创建自定义置备文件，因为这个环境变量会配置 S2I 构建过程以忽略该文件。

以下代码演示了自定义调配文件。

```
<?xml version="1.0" ?>
<installation xmlns="urn:jboss:galleon:provisioning:3.0">
    <feature-pack location="eap-s2i@maven(org.jboss.universe:s2i-universe)">(1)
```

```
    <default-configs inherit="false"/>2
    <packages inherit="false"/>3
  </feature-pack>
  <feature-pack location="org.example.mariadb:mariadb-galleon-pack:1.0-SNAPSHOT">4
    <default-configs inherit="false"/>
    <packages inherit="false"/>
  </feature-pack>
  <config model="standalone" name="standalone.xml">5
    <layers>
      <include name="jaxrs-server"/>
      <include name="mariadb-datasource"/>
    </layers>
  </config>
  <options>6
    <option name="optional-packages" value="passive+"/>
  </options>
</installation>
```

1. 此元素指示调配流程调配当前的 eap-s2i 功能包。请注意，构建器镜像仅包含一个功能包。
2. 此元素指示调配流程排除默认配置。
3. 此元素指示调配流程排除默认软件包。
4. 此元素指示调配过程调配 **org.example.mariadb:mariadb-galleon-pack:1.0-SNAPSHOT** 功能包。子元素指示进程排除默认配置和默认软件包。
5. 此元素指示调配流程创建自定义单机配置。配置包括来自 **org.example.mariadb:mariadb-galleon-pack:1.0-SNAPSHOT** 功能包的 **jaxrs-server** 基础层和 **mariadb-datasource** 自定义层。
6. 此元素指示调配流程以优化 JBoss EAP 模块的调配。

其他资源

- 有关使用 **GALLEON_PROVISION_LAYERS** 环境变量的更多信息，请参阅 调配自定义 JBoss EAP 服务器。

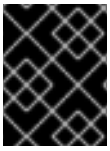
5.3.2. 使用高级环境变量配置 Galleon

您可以使用高级自定义 Galleon 功能软件包环境变量来自定义在 S2I 镜像构建过程中存储自定义 Galleon 功能软件包和层的位置。这些高级自定义 Galleon 功能软件包环境变量如下：

- **GALLEON_DIR=<path>**，它将覆盖默认的 **<project_root_dir>/galleon** 目录路径到 **<project_root_dir>/<GALLEON_DIR>**。
- **GALLEON_CUSTOM_FEATURE_PACKS_MAVEN_REPO=<path>** 会覆盖 **<project root dir>/galleon/repository** 目录路径，其绝对路径为 Maven 本地存储库缓存目录。此软件仓库包含自定义 Galleon 功能软件包。

您必须在与 Maven local-cache 文件系统配置匹配的子目录中找到 Galleon 功能软件包存档文件。例如，在 **path-to-repository/org/ examples/my-feature-pack/1.0.0.Final/my-feature-pack/1.0.0.Final/my-feature-pack-1.0.0.Final/my-feature-pack-1.0.0.Final.zip 路径中找到 org.examples: my-feature-pack-1.0.0**。

您可以通过在 **<project_root>/<GALLEON_DIR>** 目录中创建一个 **settings.xml** 文件来配置 Maven 项目设置。**GALLEON_DIR** 的默认值为 **<project_root_dir>/galleon**。Maven 使用 文件为应用程序置备自定义 Galleon 功能软件包。如果您没有创建 **settings.xml** 文件，Maven 将使用由 S2I 镜像创建的默认设置.xml 文件。



重要

不要在 **settings.xml** 文件中指定本地 Maven 存储库位置，因为 S2I 构建器镜像指定了本地 Maven 存储库的位置。S2I 构建器镜像在 S2I 构建过程中使用此位置。

其他资源

- 有关自定义 Galleon 功能软件包环境变量的更多信息，请参阅自定义 Galleon 功能软件包环境变量。

5.3.3. 自定义 Galleon 功能软件包环境变量

您可以使用以下自定义 Galleon 功能软件包环境变量来自定义如何使用 JBoss EAP S2I 镜像。

表 5.1. 自定义 Galleon 功能软件包环境变量描述

环境变量	描述
GALLEON_DIR=<path>	其中 *<path>* 是相对于应用程序项目根目录的目录。您的 *<path>* 目录包含您可选的 Galleon 自定义内容，如 **settings.xml** 文件和本地 Maven 存储库缓存。这个缓存包含自定义 Galleon 功能软件包。 目录默认为 **galleon**。
GALLEON_CUSTOM_FEATURE_PACKS_MAVEN_REPO=<path>	*<path>* 是包含自定义功能软件包的 Maven 本地存储库目录的绝对路径。目录默认为 **galleon/repository**。
GALLEON_PROVISION_FEATURE_PACKS=<list_of_galleon_feature_packs>	其中 *<list_of_galleon_feature_packs>* 是由 Maven 协调标识的自定义 Galleon 功能包的逗号分隔列表。列出的功能包必须与构建器镜像中存在的 JBoss EAP 7.4 服务器的版本兼容。 您可以使用 **GALLEON_PROVISION_LAYERS** 环境变量来设置自定义功能包定义的 Galleon 层。

第 6 章 故障排除

6.1. POD 重启故障排除

由于多种原因，容器集可以重新启动，但 JBoss EAP pod 重新启动的常见原因可能包括 OpenShift 资源约束，特别是内存不足问题。如需有关 OpenShift 容器集驱除的更多信息，请参阅 OpenShift 文档。

默认情况下，用于 OpenShift 模板的 JBoss EAP 配置为在其遇到内存不足问题等情况时自动重新启动受影响的容器。以下步骤可以帮助您诊断内存不足和其他 pod 重启问题并进行故障排除。

1. 获取遇到问题的 pod 的名称。
   您可以看到容器集名称，以及每个容器集通过以下命令重启的次数。

```
$ oc get pods
```

2. 若要诊断容器集重新启动的原因，您可以检查上一容器集的 JBoss EAP 日志或 OpenShift 事件。
   a. 要查看上述 pod 的 JBoss EAP 日志，请使用以下命令：

```
oc logs --previous POD_NAME
```

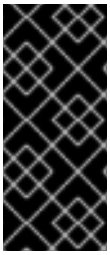
   b. 若要查看 OpenShift 事件，可使用以下命令：

```
$ oc get events
```

3. 如果容器集因为资源问题而重启，您可以尝试修改 OpenShift 容器集配置，以增加其资源请求和限值。如需有关配置 pod 计算资源的更多信息，请参阅 OpenShift 文档。

6.2. 使用 JBOSS EAP 管理 CLI 进行故障排除

JBoss EAP 管理 CLI ***EAP_HOME*/bin/jboss-cli.sh** 可以从容器内访问，以满足故障排除需要。



重要

不建议使用 JBoss EAP 管理 CLI 在运行的容器集中进行配置更改。当容器重启时，使用管理 CLI 在运行的容器中进行的任何配置更改都将丢失。

若要对 OpenShift 的 JBoss EAP 进行配置更改，请参阅为 Java 应用配置用于 OpenShift 镜像的 JBoss EAP。

1. 首先，打开一个连接至正在运行的容器集的远程 shell 会话。

```
$ oc rsh POD_NAME
```

2. 在远程 shell 会话中运行以下命令来启动 JBoss EAP 管理 CLI：

```
$ /opt/eap/bin/jboss-cli.sh
```

第 7 章 用于在 OPENSHIFT 上自动化应用部署的 EAP OPERATOR

EAP 操作器是特定于 JBoss EAP 的控制器，用于扩展 OpenShift API。您可以使用 EAP 操作器创建、配置、管理和无缝升级复杂有状态应用的实例。

EAP 操作器在集群中管理多个 JBoss EAP Java 应用实例。它还通过验证所有事务是否在缩减副本前完成，并将 pod 标记为 **干净** 以进行终止，以此确保应用集群中的安全交易恢复。EAP 操作器使用 **StatefulSet** 来适当处理 Jakarta Enterprise Beans 复制和事务恢复处理。 **StatefulSet 确保** 持久性存储和网络主机名的稳定性，即使在 pod 重启后也是如此。

您必须使用 OperatorHub 安装 EAP 操作器，供 OpenShift 集群管理员用来发现、安装和升级操作器。

在 OpenShift Container Platform 4 中，您可以使用 Operator Lifecycle Manager(OLM)来安装、更新和管理所有操作器以及在多个集群中运行的关联服务的生命周期。

OLM 在 OpenShift Container Platform 4 中默认运行。它协助集群管理员对集群上运行的操作器进行安装、升级和授予访问权限。OpenShift Container Platform Web 控制台为集群管理员提供安装操作器的管理屏幕，以及授予特定项目的访问权限，以使用集群上可用的操作器目录。

如需有关操作器和 OLM 的更多信息，请参阅 OpenShift 文档。

7.1. 使用 WEB 控制台安装 EAP OPERATOR

作为 JBoss EAP 集群管理员，您可以使用 OpenShift Container Platform Web 控制台从红帽 OperatorHub 安装 EAP operator。然后，您可以将 EAP 操作器订阅到一个或多个命名空间，供集群上的开发人员使用。

以下是在使用 Web 控制台安装 EAP 操作器前必须注意的几个要点：

- Installation Mode：选择 **All namespaces on the cluster(default)** 来在所有命名空间中安装 Operator，或选择单独的命名空间（如果可用），以仅在所选命名空间中安装 Operator。
- 更新频道：如果 EAP 操作器可以通过多个频道获得，您可以选择您要订阅的频道。例如，要通过 **stable** 频道部署（如果可用），则从列表中选择这个选项。
- **批准策略：您可以选择自动或手动更新。如果选择自动更新 EAP Operator，则当有新版本的 Operator 可用时，Operator Lifecycle Manager(OLM)会自动升级 EAP operator 的正在运行的实例。如果选择手动更新，则当有新版本的 Operator 可用时，OLM 会创建更新请求。然后，您必须手动批准更新请求，使 Operator 更新至新版本。**

注意

以下流程可能会根据 OpenShift Container Platform Web 控制台中的修改而更改。有关最新和最准确的步骤，请参阅 *OpenShift Container Platform 指南使用 Operator 的最新版本中的使用 Web 控制台从 OperatorHub 安装。*

先决条件

- **使用具有 cluster-admin 权限的账户访问 OpenShift Container Platform 集群。**

流程

1. **在 OpenShift Container Platform Web 控制台中，导航到 Operators→ OperatorHub。**
2. **向下滚动，或在 Filter by keyword 框中键入 EAP，以查找 EAP 操作器。**

3. 选择 JBoss EAP 操作器并单击 Install。

4. 在 Create Operator Subscription 页面：

   a. 任选以下一项：

      - All namespaces on the cluster(default)，将 Operator 安装至默认的 openshift-operators 命名空间，以便供集群中的所有命名空间监视和使用。该选项并非始终可用。
      - 集群上的特定命名空间会在您选择的特定单一命名空间中安装 Operator。Operator 仅在这个单一命名空间中可用。

   b. 选择一个 Update Channel。

   c. 如前面所述，选择自动或手动批准策略。

5. 单击 Subscribe，使 EAP 操作器可供 OpenShift Container Platform 集群上的所选命名空间使用。

   a. 如果您选择了手动批准策略，订阅的升级状态将保持在 Upgrading，直到您审核并批准了安装计划。在 Install Plan 页面批准安装计划后，订阅升级状态将变为 Up to date。

   b. 如果您选择了自动批准策略，则升级状态会在无需干预的情况下变为 Up to date。

6. 在订阅的升级状态变为 Up to date 后，选择 Operators → Installed Operators 来验证 EAP ClusterServiceVersion(CSV)是否显示，并在相关命名空间中显示其 Status 更改为 InstallSucceeded。

注意

对于 All namespaces... 安装模式，openshift-operators 命名空间中显示的状态为 InstallSucceeded。在其他命名空间中，显示的状态为 Copied。

7. 如果 Status 字段没有更改为 InstallSucceeded，请检查 openshift-operators 项目（如果选择了其他相关命名空间）中的 pod 的日志，这会在 Workloads → Pods 页面中报告问题以便进一步排除故障。

7.2. 使用 CLI 安装 EAP OPERATOR

作为 JBoss EAP 集群管理员，您可以使用 OpenShift Container Platform CLI 安装来自红帽 OperatorHub 的 EAP operator。然后，您可以将 EAP 操作器订阅到一个或多个命名空间，供集群上的开发人员使用。

使用 CLI 从 OperatorHub 安装 EAP operator 时，请使用 oc 命令创建 Subscription 对象。

先决条件

- 可以使用具有 cluster-admin 权限的账户访问 OpenShift Container Platform 集群。
- 已在本地系统中安装了 oc 工具。

流程

1. 查看 OperatorHub 中集群可用的操作器列表：

```
$ oc get packagemanifests -n openshift-marketplace | grep eap
NAME        CATALOG             AGE
...
eap         Red Hat Operators   43d
...
```

2. 创建一个 **Subscription** 对象 YAML 文件（如**eap-operator-sub.yaml**），以便为 EAP operator 订阅命名空间。以下是 **Subscription** 对象 YAML 文件示例：

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: eap
  namespace: openshift-operators
spec:
  channel: stable
  installPlanApproval: Automatic
  name: eap (1)
  source:  redhat-operators (2)
  sourceNamespace: openshift-marketplace
```

(1) 要订阅的操作器的名称。

(2) EAP Operator 由 **redhat-operators** CatalogSource 提供。

有关频道和批准策略的详情，请查看此流程的 Web 控制台版本。

3. 从 YAML 文件创建 **Subscription** 对象：

```
$ oc apply -f eap-operator-sub.yaml
$ oc get csv -n openshift-operators
NAME                DISPLAY     VERSION   REPLACES   PHASE
eap-operator.v1.0.0   JBoss EAP   1.0.0               Succeeded
```

EAP 操作器已安装成功。此时，OLM 已了解 EAP 操作器。Operator 的 ClusterServiceVersion(CSV)会出现在目标命名空间中，EAP 操作器提供的 API 可用于创建。

7.3. 用于创建应用镜像的 EAP-S2I-BUILD 模板

使用 **eap-s2i-build** 模板来创建应用映像。**eap-s2i-build** 模板添加多个参数，以配置应用源存储库的位置，以及用于构建应用的 EAP S2I 镜像。

eap-s2i-build 模板中的 **APPLICATION_IMAGE** 参数指定与应用镜像对应的镜像流的名称。例如，如果您从 **eap-s2i -build** 模板创建了名为 **my- app** 的应用程序镜像，您可以使用 **my-app** 镜像流中的 **my-app:latest** imagestreamtag 来部署应用。如需有关 **eap-s2i-build** 模板中使用的参数的更多信息，请参阅使用 eap-s2i-build 模板构建应用映像。

使用此模板时，EAP 操作员可以无缝升级 OpenShift 上部署的应用。要启用无缝升级，您必须在 GitHub 存储库中配置 webhook，并在构建配置中指定 webhook。当您的存储库更新并且触发新构建时，webhook 会通知 OpenShift。

您可以使用此模板来使用任何 JBoss EAP 版本（如 JBoss EAP 7.4、JBoss EAP XP 或 JBoss EAP CD）的镜像流来构建应用镜像。

其他资源

- 使用 eap-s2i-build 模板构建应用映像.

7.4. 使用 EAP-S2I-BUILD 模板构建应用程序镜像

eap-s2i-build 模板添加多个参数，以配置用于构建应用的应用源存储库的位置和 EAP S2I 镜像。通过此模板，您可以将镜像流用于任何 JBoss EAP 版本，如 JBoss EAP 7.4、JBoss EAP XP 或 JBoss EAP CD。

流程

1. **在 OpenShift 中导入 EAP 映像。如需更多信息，请参阅 导入 JBoss EAP XP 的 OpenShift 镜像流和模板。**
2. **配置镜像流以接收关于应用镜像流更改和触发新构建的更新。如需更多信息，请参阅配置定期导入 imagestreamtag。**
3. **创建 eap-s2i-build 模板，以使用 EAP S2I 镜像构建应用映像：**

```
$ oc replace --force -f https://raw.githubusercontent.com/jboss-container-images/jboss-eap-openshift-templates/master/eap-s2i-build.yaml
```

此 eap-s2i-build 模板会创建两个构建配置和两个镜像流，对应于中间构建构件和最终应用镜像。

4. **使用参数处理 eap-s2i-build 模板，以创建最终应用镜像的资源。以下示例创建了应用程序镜像 my-app ：**

```
$ oc process eap-s2i-build \
  -p APPLICATION_IMAGE=my-app \ 1
  \
  -p EAP_IMAGE=jboss-eap-xp1-openjdk11-openshift:1.0 \ 2
  -p EAP_RUNTIME_IMAGE=jboss-eap-xp1-openjdk11-runtime-openshift:1.0 \ 3
  -p EAP_IMAGESTREAM_NAMESPACE=$(oc project -q) \ 4
  \
  -p SOURCE_REPOSITORY_URL=https://github.com/jboss-developer/jboss-eap-quickstarts.git \ 5
  -p SOURCE_REPOSITORY_REF=xp-1.0.x \ 6
  -p CONTEXT_DIR=microprofile-config | oc create -f - 7
```

1. **应用程序镜像流的名称。应用镜像带有 latest 标签。**
2. **EAP 构建器镜像的 imagestreamtag。**
3. **EAP 运行时镜像的 imagestreamtag。**
4. **安装 Red Hat Middleware 镜像的镜像流的命名空间。如果省略，则使用 openshift 命名空间。只有在 openshift 以外的命名空间中安装了镜像流时，才修改此项。**
5. **应用的 Git 源 URL。**
6. **Git 分支或标签引用**
7. **包含要构建的应用的 Git 存储库中的路径。**

5. **使用 EAP 操作器准备应用镜像以进行部署。**

   a. **配置 WildFlyServer 资源：**

   ```
   $ cat > my-app.yaml<<EOF

   apiVersion: wildfly.org/v1alpha1
   kind: WildFlyServer
   metadata:
     name: my-app
   spec:
    applicationImage: 'my-app:latest'
    replicas: 1
   EOF
   ```

   b. **应用这些设置，并让 EAP 操作器创建一个新的 WildFlyServer 资源来引用此应用程序镜像：**

   ```
   $ oc apply -f my-app.yaml
   ```

   c. **使用以下命令查看 WildFlyServer 资源：**

   ```
   $ oc get wfly my-app
   ```

   其他资源

   - **有关导入应用程序镜像流的更多信息，请参阅 导入 JBoss EAP XP 的最新 OpenShift 镜像流和模板。**
   - **如需有关定期导入镜像流的更多信息，请参阅配置定期导入 imagestreamtag。**

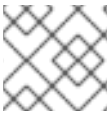
7.5. 使用 EAP OPERATOR 在 OPENSHIFT 上部署 JAVA 应用程序

EAP 操作器有助于在 OpenShift 上自动化 Java 应用部署。如需有关 EAP Operator API 的信息，请参阅 EAP Operator: API Information。

先决条件

- **已安装 EAP operator。如需有关安装 EAP 操作器的更多信息，请参阅使用 Webconsole 和 CLI 安装 EAP Operator。**
- **您已使用 JBoss EAP for OpenShift Source-to-Image(S2I)构建镜像构建了用户应用的 Docker 镜像。**
- **如果要在 OpenShift 上部署应用后启用自动升级，则 eap-s2i-build 模板中的 APPLICATION_IMAGE 参数包含镜像流。有关使用 eap-s2i-build 模板构建应用镜像的更多信息，请参阅使用 eap- s2i-build 模板构建应用镜像。**
- **如果应用的 CustomResourceDefinition(CRD)文件引用了一个对象，则已创建了 Secret 对象。有关创建新 Secret 对象的更多信息，请参阅 创建 Secret。**
- **如果应用程序的 CRD 文件引用了 ConfigMap，则创建了 ConfigMap。有关创建 ConfigMap 的详情，请参阅 创建 ConfigMap。**

- 如果您选择从 standalone.xml 文件创建 ConfigMap，请从该文件创建 ConfigMap。有关从 standalone.xml 文件创建 ConfigMap 的详情，请参考 从 standalone.xml 文件创建 ConfigMap。



注意

JBoss EAP 7 不支持从 ConfigMap 提供 standalone.xml 文件。

流程

1. 打开 Web 浏览器并登录到 OperatorHub。
2. 选择您要用于 Java 应用程序的项目或命名空间。
3. 导航到 Installed Operator，再选择 JBoss EAP operator。
4. 在 Overview 选项卡上，单击 Create Instance 链接。
5. 指定应用程序镜像详情。
   应用镜像指定包含 Java 应用的 Docker 镜像。镜像必须使用 JBoss EAP for OpenShift Source-to-Image(S2I)构建镜像进行构建。如果 applicationImage 字段与 imagestreamtag 对应，对镜像的任何更改都会触发应用程序的自动升级。

   您可以提供以下对 OpenShift 应用镜像的 JBoss EAP 的以下引用：

   - 镜像名称：mycomp/myapp
   - 标签：mycomp/myapp:1.0
   - 摘要：
     mycomp/myapp:@sha256:0af38bc38be93116b6a1d86a9c78bd14cd527121970899d719ba
   - imagestreamtag: my-app:latest
6. 指定应用程序的大小。例如：

```
spec:
  replicas:2
```

7. 使用 env spec 配置应用程序环境。环境变量 可以直接来自值，如 POSTGRESQL_SERVICE_HOST 或 Secret 对象，如 POSTGRESQL_USER。例如：

```
spec:
  env:
  - name: POSTGRESQL_SERVICE_HOST
    value: postgresql
  - name: POSTGRESQL_SERVICE_PORT
    value: '5432'
  - name: POSTGRESQL_DATABASE
    valueFrom:
      secretKeyRef:
        key: database-name
        name: postgresql
  - name: POSTGRESQL_USER
    valueFrom:
      secretKeyRef:
```

```
      key: database-user
      name: postgresql
- name: POSTGRESQL_PASSWORD
  valueFrom:
    secretKeyRef:
      key: database-password
      name: postgresql
```

8. **完成与应用程序部署相关的以下可选配置：**

- **指定服务器数据目录的存储要求。如需更多信息，请参阅为应用程序配置持久性存储。**
- **指定您在 WildFlyServerSpec 中创建的 Secret 名称，将其挂载为运行应用程序的 Pod 中的卷。例如：**

```
spec:
  secrets:
    - my-secret
```

Secret 挂载于 /etc/secrets/<secret name>，每个键/值都保存为一个文件。文件的名称是键，内容是值。Secret 作为 pod 中的卷挂载。以下示例演示了可用于查找键值的命令：

```
$ ls /etc/secrets/my-secret/
my-key  my-password
$ cat /etc/secrets/my-secret/my-key
devuser
$ cat /etc/secrets/my-secret/my-password
my-very-secure-pasword
```

注意

修改 Secret 对象可能会导致项目不一致。红帽不修改现有的 Secret 对象，而是建议创建一个内容与旧对象相同的新对象。然后，您可以根据需要更新内容，并将 operator 自定义资源(CR)中的引用从旧改为 new。这被视为一个新的 CR 更新，pod 会被重新载入。

- **指定您在 WildFlyServerSpec 中创建的 ConfigMap 名称，将其挂载为运行应用程序的 Pod 中的卷。例如：**

```
spec:
  configMaps:
  - my-config
```

ConfigMap 挂载于 /etc/configmaps/<configmap name>，每个键/值都存储为一个文件。文件的名称是键，内容是值。ConfigMap 挂载为 pod 中的卷。查找键值：

```
$ ls /etc/configmaps/my-config/
key1 key2
$ cat /etc/configmaps/my-config/key1
value1
$ cat /etc/configmaps/my-config/key2
value2
```

注意

修改 ConfigMap 可能会导致项目不一致。红帽建议创建一个与旧 ConfigMap 相同内容的新 ConfigMap，而不是修改现有的 ConfigMap。然后，您可以根据需要更新内容，并将 operator 自定义资源(CR)中的引用从旧改为 new。这被视为一个新的 CR 更新，pod 会被重新载入。

- **如果您选择有自己的独立 ConfigMap，请提供 ConfigMap 的名称以及 standalone.xml 文件的键：**

```
standaloneConfigMap:
  name: clusterbench-config-map
  key: standalone-openshift.xml
```

注意

JBoss EAP 7 不支持从 standalone.xml 文件创建 ConfigMap。

- **如果要禁用 OpenShift 中创建默认 HTTP 路由，请将 disableHTTPRoute 设置为 true ：**

```
spec:
  disableHTTPRoute: true
```

7.5.1. 创建 Secret

如果应用的 CustomResourceDefinition(CRD)文件引用了 Secret，您必须先创建 Secret，然后才能使用 EAP 操作器将应用部署到 OpenShift 中。

流程

- **创建 Secret ：**

```
$ oc create secret generic my-secret --from-literal=my-key=devuser --from-literal=my-password='my-very-secure-pasword'
```

7.5.2. 创建 ConfigMap

如果应用程序的 CustomResourceDefinition(CRD)文件引用 spec.ConfigMaps 字段中的 ConfigMap，您必须先创建 ConfigMap，然后才能使用 EAP 操作器在 OpenShift 中部署应用程序。

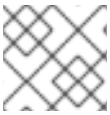
流程

- **创建 configmap：**

```
 $ oc create configmap my-config --from-literal=key1=value1 --from-literal=key2=value2
configmap/my-config created
```

7.5.3. 从 standalone.xml 文件创建 ConfigMap

您可以创建自己的 JBoss EAP 单机配置，而不使用来自 JBoss EAP for OpenShift Source-to-Image(S2I)的应用镜像中的配置。standalone.xml 文件必须放在可由 Operator 访问的 ConfigMap 中。



注意

注意：JBoss EAP 7 不支持从 ConfigMap 提供 standalone.xml 文件。

流程

- **从 standalone.xml 文件创建 ConfigMap ：**

```
 $ oc create configmap clusterbench-config-map --from-file examples/clustering/config/standalone-
openshift.xml
configmap/clusterbench-config-map created
```

7.5.4. 为应用程序配置持久性存储

如果您的应用程序需要对一些数据进行持久性存储，如在 pod 重启后必须保留的事务或消息传递日志，请配置存储规格。如果存储 spec 为空，应用程序的每个 pod 都会使用一个 EmptyDir 卷。但是，此卷在对应的 pod 停止后不会保留。

流程

1. **指定 volumeClaimTemplate，以配置资源要求，以存储 JBoss EAP 单机数据目录。模板的名称派生自 JBoss EAP 的名称。对应的卷被挂载为 ReadWriteOnce 访问模式。**

```
spec:
  storage:
    volumeClaimTemplate:
      spec:
        resources:
          requests:
            storage: 3Gi
```

满足此存储要求的持久卷挂载到 /eap/standalone/data 目录。

7.6. 使用 EAP OPERATOR 部署启用了 RED HAT SINGLE SIGN-ON 的镜像

EAP 操作器可帮助您部署在 OpenShift 上启用的红帽单点登录的 EAP 应用镜像。要部署应用程序镜像，请配置表中列出 的环境变量和 secret。

先决条件

- **已安装 EAP 操作器。有关安装 EAP 操作器的更多信息，请参阅使用 Web 控制台安装 EAP 操作器 和 使用 CLI 安装 EAP 操作器。**
- **您已使用 eap74-sso-s2i 模板构建 EAP 应用程序镜像。有关构建 EAP 应用镜像的详情，请参考 构建应用程序镜像。**

流程

1. **从构建 EAP 应用镜像的位置中删除 eap74-sso-s2i 模板创建的 DeploymentConfig 文件。**
2. **在 EAP 操作器的 WildFlyServer 资源的 env 字段中，配置 所有环境变量和机密。**

   配置示例

```
$ cat > my-app.yaml<<EOF

apiVersion: wildfly.org/v1alpha1
kind: WildFlyServer
metadata:
  name: my-app
spec:
  applicationImage: 'my-app:latest'
  replicas: 1

  env:
  - name: SSO_URL
    value: https://secure-sso-sso-app-demo.openshift32.example.com/auth
  - name: SSO_REALM
    value: eap-demo
  - name: SSO_PUBLIC_KEY
    value: realm-public-key
  - name: SSO_USERNAME
    value: mySsoUser
  - name: SSO_PASSWORD
    value: 6fedmL3P
  - name: SSO_SAML_KEYSTORE
    value: /etc/secret/sso-app-secret/keystore.jks
  - name: SSO_SAML_KEYSTORE_PASSWORD
    value: mykeystorepass
  - name: SSO_SAML_CERTIFICATE_NAME
    value: jboss
  - name: SSO_BEARER_ONLY
    value: true
  - name: SSO_CLIENT
    value: module-name
  - name: SSO_ENABLE_CORS
    value: true
  - name: SSO_SECRET
    value: KZ1QyIq4
  - name: SSO_DISABLE_SSL_CERTIFICATE_VALIDATION
    value: true
  - name: SSO_SAML_KEYSTORE_SECRET
    value: sso-app-secret
  - name: HTTPS_SECRET
    value: eap-ssl-secret
  - name: SSO_TRUSTSTORE_SECRET
    value: sso-app-secret
EOF
```

注意

- **确保所有环境变量和 secret 都与镜像配置匹配。**
- **参数的值 SSO_URL 因 OpenShift 集群的用户而异。**
- **EAP 操作器将机密挂载到 /etc/secret 目录中，而 eap74-sso 模板会将机密挂载到 /etc 目录中。**

3. **保存 EAP 操作器的 WildFlyServer 资源配置。**

7.7. 使用 EAP 操作器查看应用的指标

您可以使用 EAP 操作器查看 OpenShift 上部署的应用的指标。

当集群管理员在项目中启用了指标监控时，EAP 操作器会自动在 OpenShift 控制台中显示指标。

先决条件

- 集群管理员已为项目启用了监控。如需更多信息，请参阅为用户定义的项目启用监控。

流程

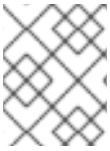
1. 在 OpenShift Container Platform Web 控制台中，导航到 Monitoring→ Metrics。
2. 在 Metrics 屏幕上，在文本框中输入应用程序名称以选择应用程序。您的应用程序的指标会出现在屏幕上。

注意

与 JBoss EAP 应用服务器相关的所有指标都以 **jboss** 为前缀。例如，**jboss_undertow_request_count_total**。

7.8. 使用 WEB 控制台卸载 EAP OPERATOR

若要从集群中删除或卸载 EAP operator，可以删除订阅，将其从订阅的命名空间中删除。您也可以移除 EAP 操作器的 ClusterServiceVersion(CSV)和部署。



注意

为确保数据一致性和安全性，请在卸载 EAP 操作器之前将集群中的 pod 数量向下扩展到 0。

您可以使用 Web 控制台卸载 EAP 操作器。

警告

如果您决定删除整个 **wildflyserver** 定义(**oc delete wildflyserver <deployment_name>**)，则不会启动事务恢复过程，无论未完成的交易都会终止 pod。此操作结果的未完成工作可能会阻止您稍后启动的数据更改。涉及与此 **wildflyserver** 交易企业 Bean 远程调用的其他 JBoss EAP 实例的数据更改也可能被阻止。

流程

1. 在 Operators→ Installed Operators 页面中，选择 JBoss EAP。
2. 在 Operator Details 页面的右侧，从 Actions 下拉菜单中选择 Uninstall Operator。

3. 如果要删除所有安装相关组件，则在看到 Remove Operator Subscription 窗口提示时，勾选 Also completely remove the Operator from the selected namespace 复选框。这会删除 CSV，从而删除与 Operator 关联的 pod、部署、自定义资源定义(CRD)和自定义资源(CR)。

4. 点击 Remove。EAP 操作器将停止运行，并且不再接收更新。

7.9. 使用 CLI 卸载 EAP OPERATOR

若要从集群中删除或卸载 EAP operator，可以删除订阅，将其从订阅的命名空间中删除。您也可以移除 EAP 操作器的 ClusterServiceVersion(CSV)和部署。

注意

为确保数据一致性和安全性，请在卸载 EAP 操作器之前将集群中的 pod 数量向下扩展到 0。

您可以使用命令行卸载 EAP 操作器。

使用命令行时，您可以通过从目标命名空间中删除订阅和 CSV 来卸载 Operator。

警告

如果您决定删除整个 wildflyserver 定义(oc delete wildflyserver <deployment_name>)，则不会启动事务恢复过程，无论未完成的交易都会终止 pod。此操作结果的未完成工作可能会阻止您稍后启动的数据更改。涉及与此 wildflyserver 交易企业 Bean 远程调用的其他 JBoss EAP 实例的数据更改也可能被阻止。

流程

1. 在 currentCSV 字段中检查 EAP operator 订阅的当前版本：

```
$ oc get subscription eap-operator -n openshift-operators -o yaml | grep currentCSV
  currentCSV: eap-operator.v1.0.0
```

2. 删除 EAP Operator 的订阅：

```
$ oc delete subscription eap-operator -n openshift-operators
subscription.operators.coreos.com "eap-operator" deleted
```

3. 使用上一步中的 currentCSV 值，在目标命名空间中删除 EAP Operator 的 CSV：

```
$ oc delete clusterserviceversion eap-operator.v1.0.0 -n openshift-operators
clusterserviceversion.operators.coreos.com "eap-operator.v1.0.0" deleted
```

7.10. 用于 SAFE 事务恢复的 EAP OPERATOR

对于特定类型的事务，EAP 操作器会在终止应用程序集群前确保数据一致性，方法是验证所有事务在缩减副本前完成，并将 pod 标记为 clean 用于终止。

注意

不支持某些场景。有关不支持的场景的更多信息，请参阅 不支持的事务恢复方案。

这意味着，如果要在没有数据不一致的情况下安全地删除部署，您必须首先将 pod 数量缩减为 0，等待所有 pod 终止，然后才删除 wildflyserver 实例。

警告

如果您决定删除整个 wildflyserver 定义(oc delete wildflyserver <deployment_name>)，则不会启动事务恢复过程，无论未完成的交易都会终止 pod。此操作结果的未完成工作可能会阻止您稍后启动的数据更改。涉及与此 wildflyserver 交易企业 Bean 远程调用的其他 JBoss EAP 实例的数据更改也可能被阻止。

当扩展过程开始 pod 状态(oc get pod <pod_name>)仍然标记为 Running，因为 pod 必须完成所有未完成事务，包括针对它的远程企业 beans 调用。

如果要监控 scaledown 进程的状态，请观察 wildflyserver 实例的状态。如需更多信息，请参阅监控缩减过程。有关扩展过程中 pod 状态的信息，请参阅 Pod 在缩减期间的状态。

7.10.1. Stable Network Host Names 的 StatefulSets

管理 wildflyserver 的 EAP 操作器将创建 StatefulSet 作为管理 JBoss EAP pod 的底层对象。

StatefulSet 是管理有状态应用程序的工作负载 API 对象。它管理一组容器集的部署和扩展，并且对这些容器集的顺序和唯一性提供保证。

StatefulSet 可确保以预定义的顺序命名集群中的 pod。它也确保 pod 终止遵循相同的顺序。例如，让我们说，pod-1 具有具有启发式结果的交易，因此处于 SCALING_DOWN_RECOVERY_DIRTY 状态。即使 pod-0 处于 SCALING_DOWN_CLEAN 状态，它也不会在 pod-1 之前终止。在 pod-1 被清理 并被终止前，pod-0 仍然处于 SCALING_DOWN_CLEAN 状态。但是，即使 pod-0 处于 SCALING_DOWN_CLEAN 状态，它也不会收到任何新的请求，实际上也没有闲置。

注意

减少 StatefulSet 的副本大小或删除 pod 本身无效，并会恢复此类更改。

7.10.2. 监控扩展过程

如果要监控 scaledown 进程的状态，您必须观察 wildflyserver 实例的状态。如需有关扩展过程中不同 pod 状态的更多信息，请参阅 Pod 在缩减期间的状态。

流程

- 观察缩减过程的状态：

```
oc describe wildflyserver <name>
```

- WildFlyServer.Status.Scalingdown Pod 和 WildFlyServer.Status.Replicas 字段显示活跃和非主动 Pod 的整体状态。
- Scalingdown Pods 字段显示所有未完成事务完成后将终止的 pod 数量。
- WildFlyServer.Status.Replicas 字段显示当前运行的容器集数量。
- WildFlyServer.Spec.Replicas 字段显示处于 ACTIVE 状态的 pod 数量。
- 如果没有缩放过程中的 pod，则 WildFlyServer.Status.Replicas 和 WildFlyServer.Spec.Replicas 字段中的 pod 数量是相等的。

7.10.2.1. 横向扩展期间的 Pod 状态

下表描述了扩展过程中的不同 pod 状态：

表 7.1. Pod 状态描述

Pod 状态	描述
ACTIVE	pod 处于活跃状态，并处理请求。
SCALING_DOWN_RECOVERY_INVESTIGATION	pod 即将缩减。纵向缩减流程正在调查 JBoss EAP 中的事务状态。
SCALING_DOWN_RECOVERY_DIRTY	JBoss EAP 包含一些不完整的交易。在清理完 pod 前，pod 不会被终止。事务恢复过程在 JBoss EAP 定期运行，它会等待事务完成
SCALING_DOWN_CLEAN	pod 通过缩减事务处理来处理，标记为 干净，可从集群中删除。

7.10.3. 使用 Heuristic Outcomes 在事务期间缩减

当交易的结果未知时，无法自动恢复交易。然后必须手动恢复您的事务。

先决条件

- pod 的状态停留在 SCALING_DOWN_RECOVERY_DIRTY。

流程

1. 使用 CLI 访问您的 JBoss EAP 实例。
2. 解决事务对象存储中的所有启发式事务记录。如需更多信息，请参阅 *JBoss EAP* 管理事务中的恢复 Heuristic Outcomes。
3. 从企业 bean 客户端恢复文件夹中删除所有记录。
   a. 从 pod enterprise bean 客户端恢复目录中删除所有文件：

```
$JBOSS_HOME/standalone/data/ejb-xa-recovery
oc exec <podname> rm -rf $JBOSS_HOME/standalone/data/ejb-xa-recovery
```

4. pod 的状态变为 **SCALING_DOWN_CLEAN**，pod 将被终止。

7.10.4. 配置事务子系统以将 JDBC 存储用于事务日志

如果系统不提供文件系统来存储 事务日志，请使用 JBoss EAP S2I 镜像配置 JDBC 对象存储。

重要

当 JBoss EAP 部署为可引导 JAR 时，S2I 环境变量不可用。在这种情况下，您必须创建 Galleon 层，或配置 CLI 脚本以进行必要的配置更改。

JDBC 对象存储可以使用环境变量 TX_DATABASE_PREFIX_MAPPING 进行设置。此变量的结构与 DB_SERVICE_PREFIX_MAPPING 相同。

前提条件

- **您已基于环境变量的值创建了数据源。**
- **您已确保数据库和通过 JDBC 对象存储通信 的交易管理器 之间存在一致的数据读取和写入权限。如需更多信息，请参阅配置 JDBC 数据源**

流程

- **通过 S2I 环境变量设置和配置 JDBC 对象存储。**

  示例

```
# Narayana JDBC objectstore configuration via s2i env variables
- name: TX_DATABASE_PREFIX_MAPPING
  value: 'PostgresJdbcObjectStore-postgresql=PG_OBJECTSTORE'
- name: POSTGRESJDBCOBJECTSTORE_POSTGRESQL_SERVICE_HOST
  value: 'postgresql'
- name: POSTGRESJDBCOBJECTSTORE_POSTGRESQL_SERVICE_PORT
  value: '5432'
- name: PG_OBJECTSTORE_JNDI
  value: 'java:jboss/datasources/PostgresJdbc'
- name: PG_OBJECTSTORE_DRIVER
  value: 'postgresql'
- name: PG_OBJECTSTORE_DATABASE
  value: 'sampledb'
- name: PG_OBJECTSTORE_USERNAME
  value: 'admin'
- name: PG_OBJECTSTORE_PASSWORD
  value: 'admin'
```

验证

- **您可以通过检查 standalone-openshift.xml 配置文件 oc rsh <podname> cat /opt/eap/standalone/configuration/standalone-openshift.xml 来验证数据源配置和事务子系统配置。**

预期输出：

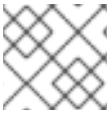
```
<datasource jta="false" jndi-name="java:jboss/datasources/PostgresJdbcObjectStore" pool-name="postgresjdbcobjectstore_postgresqlObjectStorePool"
    enabled="true" use-java-context="true" statistics-enabled="${wildfly.datasources.statistics-enabled:${wildfly.statistics-enabled:false}}">
    <connection-url>jdbc:postgresql://postgresql:5432/sampledb</connection-url>
    <driver>postgresql</driver>
    <security>
        <user-name>admin</user-name>
        <password>admin</password>
    </security>
</datasource>

<!-- under subsystem urn:jboss:domain:transactions -->
<jdbc-store datasource-jndi-name="java:jboss/datasources/PostgresJdbcObjectStore">
     <!-- the pod name was named transactions-xa-0 -->
    <action table-prefix="ostransactionsxa0"/>
    <communication table-prefix="ostransactionsxa0"/>
    <state table-prefix="ostransactionsxa0"/>
</jdbc-store>
```

其他资源

- **有关使用管理控制台或管理 CLI 创建数据源的更多信息，请参阅 JBoss EAP *配置指南中的创建数据源*。**

7.11. 使用 POD 横向自动扩展 HPA 自动扩展 POD

使用 EAP 操作器时，您可以使用 pod 横向自动扩展 HPA 根据从属于该 EAP 应用的容器集收集的指标来自动增加或减少 EAP 应用规模。



注意

使用 HPA 确保 pod 缩减时仍然会处理事务恢复。

流程

1. **配置资源：**

```
apiVersion: wildfly.org/v1alpha1
kind: WildFlyServer
metadata:
  name: eap-helloworld
spec:
  applicationImage: 'eap-helloworld:latest'
  replicas: 1
  resources:
    limits:
      cpu: 500m
      memory: 2Gi
    requests:
      cpu: 100m
      memory: 1Gi
```

重要

您必须指定 pod 中的容器的资源限值和请求，以便自动扩展才能按预期工作。

2. **创建 Horizontal pod 自动缩放器：**

```
oc autoscale wildflyserver/eap-helloworld --cpu-percent=50 --min=1 --max=10
```

验证

- **您可以通过检查副本来验证 HPA 行为。根据工作负载的增加或减少，副本数增加或降低。**

```
oc get hpa -w
NAME            REFERENCE                      TARGETS    MINPODS   MAXPODS   REPLICAS   AGE
eap-helloworld   WildFlyServer/eap-helloworld   217%/50%   1         10        1          4s
eap-helloworld   WildFlyServer/eap-helloworld   217%/50%   1         10        4          17s
eap-helloworld   WildFlyServer/eap-helloworld   133%/50%   1         10        8          32s
eap-helloworld   WildFlyServer/eap-helloworld   133%/50%   1         10        10         47s
eap-helloworld   WildFlyServer/eap-helloworld   139%/50%   1         10        10         62s
eap-helloworld   WildFlyServer/eap-helloworld   180%/50%   1         10        10         92s
eap-helloworld   WildFlyServer/eap-helloworld   133%/50%   1         10        10         2m2s
```

其他资源

- https://access.redhat.com/documentation/zh-cn/openshift_container_platform/4.10/html-single/nodes/index#nodes-pods-autoscaling

7.12. OPENSHIFT 上的 JAKARTA ENTERPRISE BEANS REMOTING

若要让 JBoss EAP 能够与 OpenShift 上的不同 JBoss EAP 集群之间的企业 Bean 远程调用正确配合，您必须了解企业 Bean 远程处理 OpenShift 上的配置选项。

注意

在 OpenShift 上部署时，请考虑使用 EAP 操作器：EAP 操作器使用 StatefulSet 来适当处理企业 Bean 远程处理和事务恢复处理。StatefulSet 确保 持久性存储和网络主机名的稳定性，即使在 pod 重启后也是如此。

当使用企业 Bean 远程调用及交易传播联系 JBoss EAP 实例时，需要保持网络主机名稳定性。即使 pod 重新启动，也必须在同一主机名下访问 JBoss EAP 实例。事务管理器是一个有状态的组件，可将持久交易数据绑定到特定的 JBoss EAP 实例。由于事务日志绑定到特定的 JBoss EAP 实例，它必须在同一实例中完成。

为防止使用 JDBC 事务日志存储时数据丢失，请确保您的数据库提供一致的数据读取和写入。当数据库横向扩展时，数据读写的一致性非常重要。

企业 bean 远程调用器有两个选项来配置远程调用：

- **定义远程出站连接。如需更多信息，请参阅配置远程出站连接。**
- **将编程 JNDI 查找用于远程服务器上的 Bean。如需更多信息，请参阅使用 Remote Jakarta 企业 Beans 客户端。**

您必须根据企业 bean 远程调用配置方法重新配置代表目标节点地址的值。

注意

远程调用的目标企业 bean 的名称必须是第一个 pod 的 DNS 地址。

StatefulSet 强制取决于 pod 的顺序。容器集按照预定义的顺序命名。例如，如果您将应用缩放为三个副本，您的容器集的名称为 eap-server-0、eap-server-1 和 eap-server-2。

EAP 操作器也使用无头服务来确保为 pod 分配特定的 DNS 主机名。如果应用使用 EAP 操作器，则会使用名称（如 eap-server-headless ）创建无头服务。在本例中，第一个容器集的 DNS 名称为 eap-server-0.eap-server-headless。

使用主机名 eap-server-0.eap-server-headless 可确保企业 bean 调用到达与集群连接的任何 EAP 实例。bootstrap 连接用于初始化 Jakarta Enterprise Beans 客户端，该客户端将 EAP 集群的结构作为下一步收集。

7.12.1. 在 OpenShift 中配置 Jakarta Enterprise Beans

您必须配置充当企业 Ban 远程调用者的 JBoss EAP 服务器。目标服务器必须配置有权接收企业 bean 远程调用的用户。

先决条件

- 您已使用 EAP 操作器和支持的 JBoss EAP for OpenShift S2I 镜像在 OpenShift 上部署和管理 JBoss EAP 应用实例。
- 集群设置正确。有关 JBoss EAP 集群的更多信息，请参阅集群部分。

流程

1. 在目标服务器中创建用户，并有权接收企业 bean 远程调用：

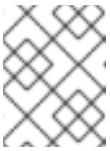
```
$JBOSS_HOME/bin/add-user.sh
```

2. 配置调用者 JBoss EAP 应用服务器。

   a. 使用自定义配置功能，在 $JBOSS_HOME/standalone/configuration 中创建 eap-config.xml 文件。如需更多信息，请参阅自定义配置。

   b. 使用 wildfly.config.url 属性配置调用者 JBoss EAP 应用服务器：

```
JAVA_OPTS_APPEND="-
Dwildfly.config.url=$JBOSS_HOME/standalone/configuration/eap-config.xml"
```



注意

如果您的配置使用以下示例，请将 >> PASTE_..._HERE<< 替换为您配置的用户名和密码。

配置示例

```
<configuration>
```

```
<authentication-client xmlns="urn:elytron:1.0">
  <authentication-rules>
    <rule use-configuration="jta">
      <match-abstract-type name="jta" authority="jboss" />
    </rule>
  </authentication-rules>
  <authentication-configurations>
    <configuration name="jta">
      <sasl-mechanism-selector selector="DIGEST-MD5" />
      <providers>
        <use-service-loader />
      </providers>
      <set-user-name name="PASTE_USER_NAME_HERE" />
      <credentials>
        <clear-password password="PASTE_PASSWORD_HERE" />
      </credentials>
      <set-mechanism-realm name="ApplicationRealm" />
    </configuration>
  </authentication-configurations>
</authentication-client>
</configuration>
```

第 8 章 参考指南

注意

本节中的内容来源于此图像的工程文档。它仅供参考，因为它可用于开发用途和超出产品文档范围进行测试。

8.1. 持久性卷

部署 JBoss EAP 和数据库 pod 的 JBoss EAP 数据库模板具有临时和持久性的变化。

永久模板包括用于调配持久卷声明的环境变量，它与可用作 OpenShift 部署的 JBoss EAP 存储卷的可用持久卷绑定。定时器架构、日志处理或数据更新等信息存储在存储卷中，而不是临时容器内存中。如果 pod 因任何原因停机，如项目升级、部署回滚或意外错误，则此信息会保留。

如果没有用于部署的永久存储卷，此信息仅存储在容器内存中，并在容器集因任何原因停机时丢失。

例如，如果容器集重启，由持久存储支持的 EE 定时器将继续运行。当应用程序再次运行时，计时器在重启过程中触发的所有事件都会被触发。

相反，如果 EE 定时器在容器内存中运行，则容器集重启后会丢失定时器状态，并在容器集再次运行时从开始开始。

8.2. 信息环境变量

以下环境变量旨在为镜像提供信息，并且不应由用户修改：

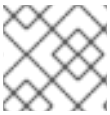
表 8.1. 信息环境变量

变量名称	描述和值
JBOSS_IMAGE_NAME	镜像名称。 值： • **jboss-eap-7/eap74-openjdk8-openshift-rhel7** (JDK 8 / RHEL 7) • **jboss-eap-7/eap74-openjdk11-openshift-rhel8** (JDK 11 / RHEL 8)
JBOSS_IMAGE_VERSION	镜像版本。 Value：这是镜像版本号。如需获得最新值，请参阅 Red Hat Container Catalog： • JDK 8 / RHEL 7 • JDK 11 / RHEL 8
JBOSS_MODULES_SYSTEM_PKGS	可供应用使用的 JBoss EAP 系统模块软件包的逗号分隔列表。 value: **jdk.nashorn.api**

变量名称	描述和值
STI_BUILDER	为 **jee** 项目类型提供 OpenShift S2I 支持。 值： **jee**

8.3. 配置环境变量

您可以配置以下环境变量来调整镜像，而无需重新构建。



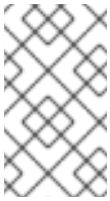
注意

有关此处未列出的其他环境变量，请参见 JBoss EAP 文档。

表 8.2. 配置环境变量

变量名称	描述
AB_JOLOKIA_AUTH_OPENSHIFT	为 OpenShift TLS 通信打开客户端身份验证。此参数的值可以是 **true**、**false** 或相对可分辨名称，该名称必须包含在所呈现的客户端证书中。默认 CA 证书被设置为 **/var/run/secrets/kubernetes.io/serviceaccount/ca.crt**。 ● 设置为 **false**，以禁用 OpenShift TLS 通信的客户端身份验证。 ● 设置为 **true**，以使用默认的 CA 证书和客户端主体为 OpenShift TLS 通信启用客户端身份验证。 ● 设置为相对可分辨的名称，如 **cn=someSystem**，为 OpenShift TLS 通信启用客户端身份验证，但会覆盖客户端主体。这种可分辨名称必须包含在所出示客户端的证书中。
AB_JOLOKIA_CONFIG	如果设置，将此完全限定的文件路径用于 Jolokia JVM 代理属性，具体如 Jolokia 参考文档中所述。如果您设置了自己的 Jolokia 属性配置文件，则忽略本文档中的其余 Jolokia 设置。 如果没有设置，**则使用 Jolokia 参考文档中定义的设置来创建 /opt/jolokia/etc/jolokia.properties**。 示例值： **/opt/jolokia/custom.properties**
AB_JOLOKIA_DISCOVERY_ENABLED	启用 Jolokia 发现。 默认值为 **false**。
AB_JOLOKIA_HOST	要绑定到的主机地址。 默认值为 **0.0.0.0**。 示例值： **127.0.0.1**

变量名称	描述
AB_JOLOKIA_HTTPS	使用 HTTPS 打开安全通信. 如果 **AB_JOLOKIA_OPTS 中没有提供 serverCert 配置，则默认生成自签名服务器证书。** 示例值： **true**
AB_JOLOKIA_ID	要使用的代理 ID。 默认值为 **$HOSTNAME**，即容器 ID。 示例值： **openjdk-app-1-xqlsj**
AB_JOLOKIA_OFF	如果设置为 **true，则** 禁用 Jolokia 的激活，这将回显空值。 默认启用 Jolokia。
AB_JOLOKIA_OPTS	要附加到代理配置的其他选项。它们应该以 **key=value, key=value, …** 格式提供。 示例值： **tasks=20**
AB_JOLOKIA_PASSWORD	基本身份验证的密码。 默认情况下关闭身份验证。 示例值： **mypassword**
AB_JOLOKIA_PASSWORD_RANDOM	确定是否应生成随机 **AB_JOLOKIA_PASSWORD**。 设置为 **true** 以生成随机密码。生成的值保存在 **/opt/jolokia/etc/jolokia.pw** 文件中。
AB_JOLOKIA_PORT	要侦听的端口。 默认值为 **8778**。 示例值： **5432**
AB_JOLOKIA_USER	用于基本身份验证的用户名称。 默认为 **jolokia**。 示例值： **myusername**

变量名称	描述
AB_PROMETHEUS_ENABLE	如果设置为 **true**，则 此变量将激活 **jmx-exporter** java 代理，该代理公开 Prometheus 格式指标。默认值为 **false**。  **注意** MicroProfile 指标子系统是以 Prometheus 格式公开数据的首选方法。如需有关 MicroProfile 指标 susbsystem 的更多信息，请参阅 JBoss EAP *配置指南中的*Eclipse MicroProfile。
AB_PROMETHEUS_JMX_EXPORTER_CONFIG	容器内的用户指定 **配置.yaml** 的路径，用于要使用的 **jmx-exporter** 代理，而不是默认的 **configuration.yaml** 文件。要了解用于合并其他配置文件的 **S2I** 机制的更多信息，请参阅 S2I 工件。
AB_PROMETHEUS_JMX_EXPORTER_PORT	**jmx-exporter** 代理在其上侦听 Prometheus 服务器提取的端口。默认值为 **9799**。代理侦听 **localhost**。通过为应用配置 **DeploymentConfig** API 使其包含此端点公开的服务，可以在容器外提供指标。
CLI_GRACEFUL_SHUTDOWN	如果设置为任何非零长度值，映像将阻止通过 **TERM 信号** 关机，并且需要使用 JBoss EAP 管理 CLI 执行 **关闭** 命令。 示例值： **true**
CONTAINER_HEAP_PERCENT	将最大 Java 堆大小设置为可用容器内存的百分比。 示例值： **0.5**
CUSTOM_INSTALL_DIRECTORIES	用于在 S2I 过程中安装和配置镜像工件的目录列表。 示例值： **custom,shared**
DEFAULT_JMS_CONNECTION_FACTORY	此值用于指定 Jakarta Messaging 连接工厂的默认 JNDI 绑定，如 **jms-connection-factory='java:jboss/DefaultJMSConnectionFactory'**。 示例值： **java:jboss/DefaultJMSConnectionFactory**

变量名称	描述
DISABLE_EMBEDDED_JMS_BROKER	在 OpenShift 容器中使用嵌入式消息传递代理已弃用。在以后的发行版本中将删除对嵌入式代理的支持。 如果满足以下条件，则会记录警告。 ● 容器配置为使用嵌入式消息传递代理。 ● 没有为容器配置远程代理。 ● 此变量未设置，或使用值 **false** 设置。 如果包含此变量的值设为 **true**，则会禁用嵌入的消息传递代理，且不会记录任何警告。 对于没有配置远程消息传递目的地的容器，请将此变量设为 **true**。
ENABLE_ACCESS_LOG	启用记录访问标准输出频道的消息。 使用以下方法实现访问信息日志记录： ● JBoss EAP 6.4 OpenShift 映像使用自定义 JBoss Web 访问日志 Valve。 ● 用于 OpenShift 镜像的 JBoss EAP 使用 Undertow **AccessLogHandler**。 默认值为 **false**。
INITIAL_HEAP_PERCENT	设置初始 Java 堆大小，作为最大堆大小的百分比。 示例值：**0.5**
JAVA_OPTS_APPEND	服务器启动选项. 示例值：**-Dfoo=bar**
JBOSS_MODULES_SYSTEM_PKGS_APPEND	以逗号分隔的软件包名称列表，附加到 **JBOSS_MODULES_SYSTEM_PKGS** 环境变量中。 示例值：**org.jboss.byteman**
JGROUPS_CLUSTER_PASSWORD	用于对节点进行身份验证的密码，以便能够加入 JGroups 群集。使用 **ASYM_ENCRYPT** JGroups 集群流量加密协议时需要。如果没有设置身份验证，则禁用身份验证，不会加密群集通信并发出警告。可选，在使用 **SYM_ENCRYPT** JGroups 集群流量加密协议时。 示例值：**mypassword**

变量名称	描述
JGROUPS_ENCRYPT_KEYSTORE	使用 **SYM_ENCRYPT JGroups 集群流量加密协议时，通过 JGROUPS_ENCRYPT_SECRYPT 变量** 指定的 secret 中的密钥存储文件名称。如果没有设置，则不会加密群集通信并发出警告。 示例值： **jgroups.jceks**
JGROUPS_ENCRYPT_KEYSTORE_DIR	使用 **SYM_ENCRYPT JGroups 集群流量加密协议时，通过 JGROUPS_ENCRYPT_ SECRET** 变量指定的 secret 中的密钥存储文件目录路径。如果没有设置，则不会加密群集通信并发出警告。 示例值： **/etc/jgroups-encrypt-secret-volume**
JGROUPS_ENCRYPT_NAME	在使用 **SYM_ENCRYPT** JGroups 集群流量加密协议时，与服务器证书关联的名称。如果没有设置，则不会加密群集通信并发出警告。 示例值： **jgroups**
JGROUPS_ENCRYPT_PASSWORD	在使用 **SYM_ENCRYPT** JGroups 集群流量加密协议时，用于访问密钥存储和证书的密码。如果没有设置，则不会加密群集通信并发出警告。 示例值： **mypassword**
JGROUPS_ENCRYPT_PROTOCOL	用于加密群集流量的 JGroups 协议.可以是 **SYM_ENCRYPT** 或 **ASYM_ENCRYPT**。 默认为 **SYM_ENCRYPT**。 示例值： **ASYM_ENCRYPT**
JGROUPS_ENCRYPT_SECRET	包含 **JGroups 密钥存储** 文件的机密名称，以便在使用 **SYM_ENCRYPT** JGroups 集群流量加密协议时保护 JGroups 通信。如果没有设置，则不会加密群集通信并发出警告。 示例值： **eap7-app-secret**
JGROUPS_PING_PROTOCOL	用于节点发现的 JGroups 协议。可以是 **dns.DNS_PING** 或 **kubernetes.KUBE_PING**。
MQ_SIMPLE_DEFAULT_PHYSICAL_DESTINATION	为向后兼容，设置为 **true 以使用 MyQueue** 和 **MyTopic** 作为物理目标名称默认值，而不使用 **queue/MyQueue** 和 **topic/MyTopic**。
OPENSHIFT_DNS_PING_SERVICE_NAME	在服务器上公开 ping 端口的服务名称，用于 DNS 发现机制。 示例值： **eap-app-ping**

变量名称	描述
OPENSHIFT_DNS_PING_SERVICE_PORT	DNS 发现机制的 ping 端口的端口号。如果没有指定，则会尝试从服务的 SRV 记录中发现端口号，否则使用默认的 **8888**。 默认值为 **8888**。
OPENSHIFT_KUBE_PING_LABELS	Kubernetes 发现机制的集群标签选择器。 示例值： **app=eap-app**
OPENSHIFT_KUBE_PING_NAMESPACE	集群用于 Kubernetes 发现机制的项目命名空间。 示例值：my **project**
SCRIPT_DEBUG	如果设置为 **true**，请确保使用 **-x** 选项执行 Bash 脚本，在命令及其执行过程中打印命令及其参数。

8.4. 应用程序模板

表 8.3. 应用程序模板

变量名称	描述
AUTO_DEPLOY_EXPLODED	控制是否应自动部署展开式部署内容。 示例值： **false**

8.5. 公开的端口

表 8.4. 公开的端口

端口号	描述
8443	HTTPS
8778	Jolokia Monitoring

8.6. DATASOURCES

根据某些环境变量的值自动创建数据源。

最重要的环境变量是 DB_SERVICE_PREFIX_MAPPING，因为它为数据源定义 JNDI 映射。这个变量允许的值是一个用逗号分开的 *POOLNAME-DATABASETYPE=PREFIX trip*lets 列表，其中：

- ***POOLNAME* 用作数据源中的 pool-name。**
- ***DATABASETYPE* 是要使用的数据库驱动程序。**

- ***PREFIX*** 是环境变量名称中用于配置数据源的前缀。

8.6.1. Datasources 的 JNDI 映射

对于 **DB_SERVICE_*PREFIX_MAPPING* 环境变量中定义的每个 *POOLNAME-DATABASETYPE= PREFIX_MAPPING* 环境变量，启动脚本会创建一个单独的数据源，在运行镜像时执行。**

注意

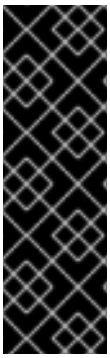
DB_SERVICE_PREFIX_MAPPING 的第一个部分（在等号前）应小写。

The ***DATABASETYPE*** 决定数据源的驱动程序。

有关配置驱动程序的更多信息，请参阅模块、驱动程序和通用部署。JDK 8 镜像默认配置了 **postgresql** 和 **mysql** 的驱动程序。

警告

不要对 *POOLNAME* 参数使用任何特殊字符。



数据库驱动程序

现在，弃用了对将红帽提供的内部数据源驱动程序与 JBoss EAP 搭配使用的支持。红帽建议您将数据库供应商获取的 JDBC 驱动程序用于 JBoss EAP 应用。

JBoss EAP for OpenShift 镜像不再提供以下内部数据源：

- MySQL
- PostgreSQL

有关安装驱动程序的更多信息，请参阅模块、驱动程序和通用部署。

有关使用 JBoss EAP 配置 JDBC 驱动程序的更多信息，请参阅 JBoss EAP *配置指南中的* JDBC 驱动程序。

请注意，如果您想将其添加到置备的服务器中，您也可以创建自定义层来安装这些驱动程序和数据源。

8.6.1.1. 数据源配置环境变量

要配置其他数据源属性，请使用以下环境变量：

重要

务必将下列变量名称中的 ***POOLNAME***、***DATABASETYPE*** 和 ***PREFIX*** 的值替换为适当的值。本节和 Datasources 部分中描述了这些可替换值。

变量名称	描述
POOLNAME _DATABASETYPE_SERVICE_HOST	定义要在数据源的 **connection-url** 属性中使用的数据库服务器的主机名或 IP 地址。 示例值：192. **168.1.3**
POOLNAME _DATABASETYPE_SERVICE_PORT	定义数据源的数据库服务器端口。 示例值： **5432**
前缀_BACKGROUND_VALIDATION	当设置为 **true** 数据库连接时，会在使用前在后台线程中定期验证数据库连接。默认为 **false**，表示默认情况下启用了 **validate-on-match** 方法。
*PREFIX*_BACKGROUND_VALIDATION_MILLIS	在启用了 **background-validation** 数据库连接验证机制时（***PREFIX*_BACKGROUND_VALIDATION** 变量设置为 **true**），以毫秒为单位指定验证频率。默认值为 **10000**。
前缀_CONNECTION_CHECKER	指定连接检查器类，用于验证正在使用的特定数据库的连接。 示例值： **org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLValidConnectionChecker**
*前缀*_DATABASE	定义数据源的数据库名称。 示例值： **myDatabase**
*前缀*_DRIVER	为数据源定义 Java 数据库驱动程序。 示例值： **postgresql**
*PREFIX*_EXCEPTION_SORTER	指定异常分类器类，用于在致命数据库连接异常后正确检测和清理。 示例值： **org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter**
*PREFIX*_JNDI	定义数据源的 JNDI 名称。默认为 **java:jboss/datasources/*POOLNAME_DATABASETYPE***，其中 ***POOLNAME*** 和 ***DATABASETYPE*** 从上面的三边图获取。如果要覆盖默认生成的 JNDI 名称，此设置很有用。 示例值： **java:jboss/datasources/test-postgresql**
*PREFIX*_JTA	为非 XA 数据源定义 Jakarta Transactions 选项。默认情况下，XA 数据源已经能够支持 Jakarta Transactions。 默认值为 **true**。

变量名称	描述
*PREFIX*_MAX_POOL_SIZE	为数据源定义最大池大小选项。 示例值： **20**
*PREFIX*_MIN_POOL_SIZE	定义数据源的最小池大小选项。 示例值 **：1**
*PREFIX*_NONXA	将数据源定义为非 XA 数据源。默认值为 **false**。
*前缀*_PASSWORD	定义数据源的密码。 示例值： **password**
*PREFIX*_TX_ISOLATION	定义数据源的 java.sql.Connection 事务隔离级别。 示例值： **TRANSACTION_READ_UNCOMMITTED**
*前缀*_URL	定义数据源的连接 URL。 示例 value: **jdbc:postgresql://localhost:5432/postgresdb**
*前缀*_USERNAME	定义数据源的用户名。 示例值： **admin**

在 OpenShift 中运行此镜像时，*POOLNAME_DATABASETYPE*_SERVICE_HOST 和 *POOLNAME_DATABASETYPE*_SERVICE_PORT 环境变量会自动从 OpenShift 应用模板中的数据库服务定义设置，而其他变量则直接在模板中配置为各个容器集模板的容器镜像 条目。

8.6.1.2. 示例

这些示例演示了 DB_SERVICE_PREFIX_MAPPING 环境变量的值如何影响数据源创建。

8.6.1.2.1. 单映射

考虑值 test-postgresql=TEST.

这将创建一个带有 java:jboss/datasources/test_postgresql 名称的数据源。此外，密码和用户名等所有必要设置都应作为环境变量提供，并附带 TEST_ 前缀，如 TEST_USERNAME 和 TEST_PASSWORD。

8.6.1.2.2. 多个映射

您可以指定多个数据源映射。

注意

始终使用逗号分隔多个数据源映射。

请考虑 DB_SERVICE_PREFIX_MAPPING 环境变量的以下值：cloud-postgresql=CLOUD,test-mysql=TEST_MYSQL。

这会创建以下两个数据源：

1. java:jboss/datasources/test_mysql
2. java:jboss/datasources/cloud_postgresql

然后，您可以使用 TEST_MYSQL 前缀来配置诸如 MySQL 数据源的用户名和密码等操作，如 TEST_MYSQL_USERNAME。对于 PostgreSQL 数据源，使用 CLOUD_ 前缀，如 CLOUD_USERNAME。

8.7. 集群

8.7.1. 配置 JGroups 发现机制

要在 OpenShift 上启用 JBoss EAP 集群，请在 JBoss EAP 配置中配置 JGroups 协议堆栈，以使用 kubernetes.KUBE_PING 或 dns.DNS_PING 发现机制。

虽然您可以使用自定义 standalone-openshift.xml 配置文件，但建议您使用环境变量 在镜像构建中配置 JGroups。

以下说明使用环境变量来配置 JBoss EAP 用于 OpenShift 镜像的发现机制。

重要

如果您使用其中一个可用的应用模板在 JBoss EAP for OpenShift 镜像基础上部署应用，则默认发现机制为 dns.DNS_PING。

dns.DNS_PING 和 kubernetes.KUBE_PING 发现机制彼此不兼容。无法从两个独立的子集群组成一个超级集群，一个使用 dns.DNS_PING 机制进行 发现，另一个使用 kubernetes.KUBE_PING 机制。同样，在执行滚动升级时，对源和目标集群的发现机制需要相同。

8.7.1.1. Configuring KUBE_PING

使用 KUBE_PING JGroups 发现机制：

1. JGroups 协议堆栈必须配置为使用 KUBE_PING 作为发现机制。
   您可以通过将 JGROUPS_PING_PROTOCOL 环境变量设置为 kubernetes.KUBE_PING 来做到这一点 ：

   ```
   JGROUPS_PING_PROTOCOL=kubernetes.KUBE_PING
   ```

2. KUBERNETES_NAMESPACE 环境变量必须设置为您的 OpenShift 项目名称。如果没有设置，服务器会作为单节点集群（"一个群集"）的行为。例如：

   ```
   KUBERNETES_NAMESPACE=PROJECT_NAME
   ```

3. 应设置 KUBERNETES_LABELS 环境变量。这应该与在服务级别设置的标签匹配。如果没有设置，则应用程序之外的 pod（您的命名空间中的某个 pod）将尝试加入。例如：

   ```
   KUBERNETES_LABELS=application=APP_NAME
   ```

4. 必须将授权授予 Pod 在 下运行的服务帐户，以允许访问 Kubernetes 的 REST API。这通过 OpenShift CLI 完成。以下示例在当前项目命名空间中使用 default 服务帐户：

```
oc policy add-role-to-user view system:serviceaccount:$(oc project -q):default -n $(oc project -q)
```

在项目命名空间中使用 eap-service-account:

```
oc policy add-role-to-user view system:serviceaccount:$(oc project -q):eap-service-account -n $(oc project -q)
```

注意

有关添加策略到服务帐户的更多信息，请参阅为应用程序部署准备 OpenShift。

8.7.1.2. 配置 DNS_PING

使用 DNS_PING JGroups 发现机制：

1. JGroups 协议堆栈必须配置为使用 DNS_PING 作为发现机制。
   您可以通过将 JGROUPS_PING_PROTOCOL 环境变量设置为 dns.DNS_PING 来做到这一点 ：

```
JGROUPS_PING_PROTOCOL=dns.DNS_PING
```

2. OPENSHIFT_DNS_PING_SERVICE_NAME 环境变量必须设置为集群的 ping 服务的名称。

```
OPENSHIFT_DNS_PING_SERVICE_NAME=PING_SERVICE_NAME
```

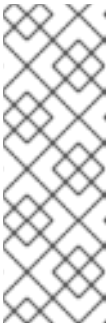
3. OPENSHIFT_DNS_PING_SERVICE_PORT 环境变量应设置为公开 ping 服务的端口号。DNS_PING 协议尝试从 SRV 记录辨别端口，否则默认为8888。

```
OPENSHIFT_DNS_PING_SERVICE_PORT=PING_PORT
```

4. 必须定义公开 ping 端口的 ping 服务。这个服务应该是无头的(ClusterIP=None)，且必须具有以下内容：

   a. 端口必须命名为。

   b. 该服务必须使用 service.alpha.kubernetes.io/tolerate-unready-endpoints 和 publishNotReadyAddresses 属性进行注解，两者都设置为 true。

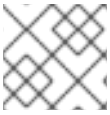


注意

- 使用 service.alpha.kubernetes.io/tolerate-unready-endpoints 和 publishNotReadyAddresses 属性来确保 ping 服务在较旧和较新的 OpenShift 版本中正常工作。
- 省略这些注解会导致每个节点在启动过程中形成自己的"集群"。然后，每个节点在启动后将其集群合并到其他节点的集群中，因为其他节点在启动后才会检测到。

```
kind: Service
```

```
apiVersion: v1
spec:
  publishNotReadyAddresses: true
  clusterIP: None
  ports:
  - name: ping
    port: 8888
  selector:
    deploymentConfig: eap-app
metadata:
  name: eap-app-ping
  annotations:
    service.alpha.kubernetes.io/tolerate-unready-endpoints: "true"
    description: "The JGroups ping port for clustering."
```



注意

DNS_PING 不需要对服务帐户进行任何修改，并使用默认权限。

8.7.2. 将 JGroups 配置为加密集群流量

若要加密 OpenShift 上 JBoss EAP 的集群流量，您必须在 JBoss EAP 配置中配置 JGroups 协议堆栈，以使用 SYM_ENCRYPT 或 ASYM_ENCRYPT 协议。

虽然您可以使用自定义 standalone-openshift.xml 配置文件，但 建议您使用环境变量 在镜像构建中配置 JGroups。

以下说明使用环境变量来配置用于 OpenShift 镜像的 JBoss EAP 集群流量加密的协议。

重要

SYM_ENCRYPT 和 ASYM_ENCRYPT 协议彼此不兼容。无法从两个独立的子集群组成一个超级集群，一个使用 SYM_ENCRYPT 协议加密集群流量，另一个使用 ASYM_ENCRYPT 协议加密。同样，在执行滚动升级时，该协议对于源和目标集群需要相同。

8.7.2.1. Configuring SYM_ENCRYPT

使用 SYM_ENCRYPT 协议加密 JGroups 集群流量：

1. JGroups 协议堆栈必须配置为使用 SYM_ENCRYPT 作为加密协议。
   您可以通过将 JGROUPS_ENCRYPT_PROTOCOL 环境变量设置为 SYM_ENCRYPT 来做到这一点 ：

   ```
   JGROUPS_ENCRYPT_PROTOCOL=SYM_ENCRYPT
   ```

2. JGROUPS_ENCRYPT_SECRET 环境变量必须设置为包含用于保护 JGroups 通信的 JGroups 密钥存储文件的机密的名称。如果没有设置，则不会加密群集通信并发出警告。例如：

   ```
   JGROUPS_ENCRYPT_SECRET=eap7-app-secret
   ```

3. JGROUPS_ENCRYPT_KEYSTORE_DIR 环境变量必须设置为通过 JGROUPS_ENCRYPT_SECRET 变量指定的机密中密钥存储文件的目录路径。如果没有设置，则不会加密群集通信并发出警告。例如：

```
JGROUPS_ENCRYPT_KEYSTORE_DIR=/etc/jgroups-encrypt-secret-volume
```

4. **JGROUPS_ENCRYPT_KEYSTORE 环境变量必须设置为通过 JGROUPS_ENCRYPT_SECRET 变量指定的机密中的密钥存储文件的名称。如果没有设置，则不会加密群集通信并发出警告。例如：**

```
JGROUPS_ENCRYPT_KEYSTORE=jgroups.jceks
```

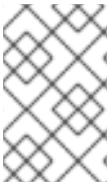
5. **JGROUPS_ENCRYPT_NAME 环境变量必须设置为与服务器证书关联的名称。如果没有设置，则不会加密群集通信并发出警告。例如：**

```
JGROUPS_ENCRYPT_NAME=jgroups
```

6. **JGROUPS_ENCRYPT_PASSWORD 环境变量必须设置为用于访问密钥存储和证书的密码。如果没有设置，则不会加密群集通信并发出警告。例如：**

```
JGROUPS_ENCRYPT_PASSWORD=mypassword
```

8.7.2.2. Configuring ASYM_ENCRYPT



注意

JBoss EAP 7.4 包括新版本的 ASYM_ENCRYPT 协议。之前版本的协议已被弃用。如果指定 JGROUPS_CLUSTER_PASSWORD 环境变量，则使用已弃用的协议版本，并在 pod 日志中显示警告。

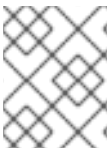
要使用 ASYM_ENCRYPT 协议加密 JGroups 集群流量，请指定 ASYM_ENCRYPT 作为加密协议，并将它配置为使用 elytron 子系统中配置的密钥存储。

```
-e JGROUPS_ENCRYPT_PROTOCOL="ASYM_ENCRYPT" \
-e JGROUPS_ENCRYPT_SECRET="encrypt_secret" \
-e JGROUPS_ENCRYPT_NAME="encrypt_name" \
-e JGROUPS_ENCRYPT_PASSWORD="encrypt_password" \
-e JGROUPS_ENCRYPT_KEYSTORE="encrypt_keystore" \
-e JGROUPS_CLUSTER_PASSWORD="cluster_password"
```

8.7.3. 扩展 pod 的注意事项

根据 JGroups 中的发现机制，启动节点将搜索现有集群协调器节点。如果在给定超时内找不到协调器节点，则起始节点会假定它是第一个成员，它会占用协调器状态。

当多个节点同时启动时，它们都假定第一个成员为第一个成员，从而导致使用多个分区的分割集群。例如，使用 DeploymentConfig API 扩展从 0 到 2 个 pod 可能会导致分割集群创建。为避免这种情况，您需要启动第一个 pod，然后再扩展至所需 pod 数量。



注意

默认情况下，EAP Operator 使用 StatefulSet API，按顺序启动 pod，即逐个启动 pod，从而防止创建拆分集群。

8.8. 健康检查

用于 OpenShift 镜像的 JBoss EAP 默认利用 OpenShift 中包含的存活度和就绪度探测。此外，此镜像还包括 Eclipse MicroProfile 健康，*如配置指南中所述*。

下表演示了这些健康检查通过所需的值。如果状态是下面找到的值以外的任何内容，则检查将失败，并且镜像按照镜像的重启策略重新启动。

表 8.5. 存活度和就绪度检查

已执行的测试	存活度	就绪
服务器状态	任何状态	Running
引导错误	无	无
部署状态 [a]	不适用或无 **失败** 条目	不适用或无 **失败** 条目
Eclipse MicroProfile 健康 [b]	无或 **UP**	无或 **UP**

[a] 仅当没有部署时，N/A 才有效。

[b] 仅当 **microprofile-health-smallrye** 子系统被禁用后，N/A 才有效状态。

8.9. 消息传递

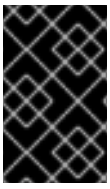
8.9.1. 配置外部 Red Hat AMQ Broker

您可以使用环境变量为 OpenShift 镜像配置 JBoss EAP，以连接外部红帽 AMQ 代理。

OpenShift 应用程序定义示例
以下示例使用模板来创建连接到外部红帽 AMQ 7 代理的 JBoss EAP 应用：

示例：JDK 8

```
oc new-app eap74-amq-s2i \
-p EAP_IMAGE_NAME=jboss-eap74-openjdk8-openshift:7.4.0 \
-p EAP_RUNTIME_IMAGE_NAME=jboss-eap74-openjdk8-runtime-openshift:7.4.0 \
-p APPLICATION_NAME=eap74-mq \
-p MQ_USERNAME=MY_USERNAME \
-p MQ_PASSWORD=MY_PASSWORD
```



重要

本例中使用的模板为所需的参数提供有效的默认值。如果不使用模板并提供自己的参数，请注意 MQ_SERVICE_PREFIX_MAPPING 名称必须与 APPLICATION_ NAME 名称匹配，并附加有 "-amq7=MQ"。

8.10. 安全域

要配置新的安全域，用户必须定义 SECDOMAIN_NAME 环境变量。

这将创建以 环境变量命名的安全域。用户也可以定义以下环境变量来自定义域：

表 8.6. 安全域

变量名称	描述
SECDOMAIN_NAME	定义一个额外的安全域。 示例值： **myDomain**
SECDOMAIN_PASSWORD_STACKING	如果定义，则启用 **password-stacking** 模块选项，并设置为值 **useFirstPass**。 示例值： **true**
SECDOMAIN_LOGIN_MODULE	要使用的登录模块。 默认为 **UsersRoles**
SECDOMAIN_USERS_PROPERTIES	包含用户定义的属性文件的名称。 默认为 **users.properties**
SECDOMAIN_ROLES_PROPERTIES	包含角色定义的属性文件的名称。 默认为 **roles.properties**

8.11. HTTPS 环境变量

变量名称	描述
HTTPS_NAME	如果与 **HTTPS_PASSWORD** 和 **HTTPS_KEYSTORE** 一起定义，请启用 HTTPS 并设置 SSL 名称。 如果您使用 **keytool -genkey** 命令创建该值，则该值应该是您的密钥存储的别名名称。 示例值： **example.com**
HTTPS_PASSWORD	如果与 **HTTPS_NAME** 和 **HTTPS_KEYSTORE** 一起定义，请启用 HTTPS 并设置 SSL 密钥密码。 示例值： **passw0rd**
HTTPS_KEYSTORE	如果与 **HTTPS_PASSWORD** 和 **HTTPS_NAME** 一起定义，请启用 HTTPS，并将 SSL 证书密钥文件设置为 ***EAP_HOME*/standalone/configuration**下的相对路径 示例值： **ssl.key**

8.12. 管理环境变量

表 8.7. 管理环境变量

变量名称	描述
ADMIN_USERNAME	如果定义了此和 **ADMIN_PASSWORD**，则用于 JBoss EAP 管理用户名。 示例值： **eapadmin**
ADMIN_PASSWORD	指定 **ADMIN_USERNAME** 的密码。 示例值： **passw0rd**

8.13. S2I

镜像包含 S2I 脚本和 Maven。

目前仅支持 Maven 作为构建工具，用于应在 OpenShift 上的基于 JBoss EAP 的容器（或相关/子镜像）上部署的应用。

目前只支持 WAR 部署。

8.13.1. 自定义配置

可以为镜像添加自定义配置文件。放入 configuration/ 目录中的所有文件将复制到 EAP_HOME/standalone/configuration/。例如，若要覆盖镜像中使用的默认配置，只需将自定义 standalone-openshift.xml 添加到 configuration/ 目录中：请参阅此类 部署的示例。

8.13.1.1. 自定义模块

可以添加自定义模块。模块/ 目录中的所有文件将复制到 EAP_HOME/modules/。请参阅此类 部署的示例。

8.13.2. deployment Artifacts

默认情况下，将部署来自源 目标 目录中的工件。要从不同目录部署，请在 BuildConfig 定义中设置 ARTIFACT_DIR 环境变量。ARTIFACT_DIR 是一个以逗号分隔的列表。例如：ART IFACT_DIR=app1/target,app2/target,app3/target

8.13.3. 工件存储库镜像

Maven 中的存储库包含各种类型的构建构件和依赖项，如所有项目 JAR、库 JAR、插件或其他特定于项目的工件。它还指定执行 S2I 构建时从哪里下载工件的位置。除了使用中央存储库外，组织通常要部署本地自定义镜像存储库。

使用镜像的好处包括：

- **同步镜像的可用性，在地理上更加接近，速度更快。**
- **能够更好地控制存储库内容。**
- **有可能在不同团队（开发人员、CI）之间共享构件，而无需依赖公共服务器和存储库。**

- 缩短构建时间.

通常，存储库管理器可以作为镜像的本地缓存。假设存储库管理器已在 https://10.0.0.1:8443/repository/internal/ 外部部署并可访问，S2I 构建可以通过向应用程序的构建配置提供 MAVEN_MIRROR_URL 环境变量来使用此管理器：

1. 识别要对其应用 MAVEN_MIRROR_URL 变量的构建配置名称。

```
oc get bc -o name
buildconfig/eap
```

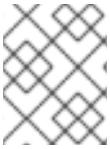
2. 使用 MAVEN_MIRROR_URL 环境变量更新 eap 的构建配置。

```
oc env bc/eap MAVEN_MIRROR_URL="https://10.0.0.1:8443/repository/internal/"
buildconfig "eap" updated
```

3. 验证 设置。

```
oc env bc/eap --list
# buildconfigs eap
MAVEN_MIRROR_URL=https://10.0.0.1:8443/repository/internal/
```

4. 计划应用的新构建。



注意

在应用构建期间，您会注意到 Maven 依赖项是从存储库管理器（而非默认的公共存储库）中提取的。另外，构建完成后，您会看到镜像已填充构建期间检索和使用的所有依赖项。

8.13.3.1. Secure Artifact Repository Mirror URL

为通过 Maven 存储库防止"man-in-the-middle"攻击，JBoss EAP 需要将安全 URL 用于工件存储库镜像 URL。

URL 应指定一个安全 http("https")和安全端口。

默认情况下，如果您指定了不安全的 URL，则会返回一个错误。您可以使用属性 -Dinsecure.repositories=WARN 来覆盖此行为。

8.13.4. 脚本

run

此脚本使用 openshift-launch.sh 脚本，通过 standalone-openshift.xml 配置配置和启动 JBoss EAP。

assemble

此脚本使用 Maven 构建来源，创建软件包(WAR)，并将它移到 EAP_HOME/standalone/deployments 目录。

8.13.5. 自定义脚本

您可以在启动 pod 之前添加自定义脚本，在 JBoss EAP 启动之前运行。

您可以添加在启动 pod 时有效的任何脚本，包括 CLI 脚本。

从镜像启动 JBoss EAP 时有两个选项可用于包括脚本：

- 将要以 postconfigure.sh 执行的 configmap 挂载
- 在指定的安装目录中添加 install.sh 脚本

8.13.5.1. 挂载 configmap 来执行自定义脚本

如果要在运行时将自定义脚本挂载到现有镜像（换句话说，是已构建的镜像），请挂载 configmap。

挂载 configmap：

1. 创建一个包含您要包含在 postconfigure.sh 中的内容的 configmap。
   例如，在项目根目录中创建一个名为 **extensions** 的目录，以包含脚本 **postconfigure.sh** 和 **extensions.cli** 并运行以下命令：

```
$ oc create configmap jboss-cli --from-file=postconfigure.sh=extensions/postconfigure.sh --
from-file=extensions.cli=extensions/extensions.cli
```

2. 通过部署控制器(dc)将 configmap 挂载到容器集中。

```
$ oc set volume dc/eap-app --add --name=jboss-cli -m /opt/eap/extensions -t configmap --
configmap-name=jboss-cli --default-mode='0755' --overwrite
```

postconfigure.sh示例

```
#!/usr/bin/env bash
set -x
echo "Executing postconfigure.sh"
$JBOSS_HOME/bin/jboss-cli.sh --file=$JBOSS_HOME/extensions/extensions.cli
```

extensions.cli示例

```
embed-server --std-out=echo  --server-config=standalone-openshift.xml
:whoami
quit
```

8.13.5.2. 使用 install.sh 执行自定义脚本

如果要在构建时包含脚本作为镜像的一部分，请使用 install.sh。

使用 install.sh 执行自定义脚本：

1. 在 s2i 构建期间要使用的项目的 git 存储库中，创建一个名为 **.s2i** 的目录。
2. 在 **s2i** 目录中，添加包含以下内容的名为 environment 的文件：

```
$ cat .s2i/environment
CUSTOM_INSTALL_DIRECTORIES=extensions
```

3. 创建名为扩展名 的目录。

4. *在 扩展 目录中，创建包含以下内容的 postconfigure.sh 文件（将占位符代码替换为您的环境的适当代码）：*

```
$ cat extensions/postconfigure.sh
#!/usr/bin/env bash
echo "Executing patch.cli"
$JBOSS_HOME/bin/jboss-cli.sh --file=$JBOSS_HOME/extensions/some-cli-example.cli
```

5. *在扩展目录中，创建类似于以下内容的文件 install.sh（用适合您的环境替换占位符代码）：*

```
$ cat extensions/install.sh
#!/usr/bin/env bash
set -x
echo "Running $PWD/install.sh"
injected_dir=$1
# copy any needed files into the target build.
cp -rf ${injected_dir} $JBOSS_HOME/extensions
```

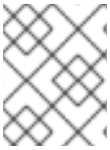
8.13.6. 环境变量

您可以通过向 s2i build 命令提供环境变量来影响构建的执行方式。可以提供的环境变量有：

表 8.8. s2i 环境变量

变量名称	描述
ARTIFACT_DIR	此目录中的 **.war**、**. ear 和.jar** 文件将复制到 **deployments/** 目录中。 示例值： **target**
ENABLE_GENERATE_DEFAULT_DATASOURCE	可选。当带有值 **true** 时，服务器会被置备为 default 数据源。否则，默认数据源不包含在内。
GALLEON_PROVISION_DEFAULT_FAT_SERVER	可选。如果值为 **true**，并且未设置 galleon 层，则调配了一个默认的 JBoss EAP 服务器。
GALLEON_PROVISION_LAYERS	可选。指示 S2I 进程调配指定的层。该值是一个要调配的以逗号分隔的层列表，包括一个基础层和任意数量的 decorator 层。 示例值： **jaxrs、sso**
HTTP_PROXY_HOST	用于 Maven 的 HTTP 代理的主机名或 IP 地址。 示例值：192. **168.1.1**
HTTP_PROXY_PORT	Maven 使用的 HTTP 代理的 TCP 端口。 示例值：80 **80**

变量名称	描述
HTTP_PROXY_USERNAME	如果提供了 **HTTP_PROXY_PASSWORD**，请将凭证用于 HTTP 代理。 示例值： **myusername**
HTTP_PROXY_PASSWORD	如果提供了 **HTTP_PROXY_USERNAME**，请对 HTTP 代理使用凭据。 示例值： **mypassword**
HTTP_PROXY_NONPROXYHOSTS	如果提供，配置的 HTTP 代理将忽略这些主机。 示例值： **some.example.org|*.example.net**
MAVEN_ARGS	覆盖构建期间提供给 Maven 的参数。 示例值： **-e -Popenshift -DskipTests -Dcom.redhat.xpaas.repo.redhatga 软件包**
MAVEN_ARGS_APPEND	在构建期间将用户参数附加到 Maven。 示例值： **-Dfoo=bar**
MAVEN_MIRROR_URL	要配置的 Maven Mirror/repository Manager 的 URL。 示例值： **https://10.0.0.1:8443/repository/internal/** 请注意，指定的 URL 应该安全。详情请查看 第 8.13.3.1 节 “Secure Artifact Repository Mirror URL”。
MAVEN_CLEAR_REPO	（可选）在构建后清除本地 Maven 存储库。 如果镜像中的服务器与本地缓存高度耦合，则不会删除缓存并打印警告。 示例值： **true**
APP_DATADIR	如果定义，则在复制数据文件的源中的目录。 示例值： **mydata**
DATA_DIR	镜像中复制 **$APP_DATADIR** 数据的目录。 示例值： ***EAP_HOME*/data**



注意

如需更多信息，请参阅在用于 OpenShift 镜像的 JBoss EAP 上构建和运行 Java 应用，该镜像使用 Maven 和 JBoss EAP 中为 OpenShift 镜像中包含的 S2I 脚本。

8.14. 单点登录镜像

此镜像包括启用了红帽单点登录的应用程序。

有关使用 OpenShift 镜像的 JBoss EAP 镜像部署 Red Hat Single Sign-On for OpenShift镜像的更多信息，请参阅在 Red Hat Single Sign-On for OpenShift 指南上部署 Red Hat Single Sign-On 的 JBoss EAP 镜像。

表 8.9. 单点登录环境变量

变量名称	描述
SSO_URL	单点登录服务器的 URL。
SSO_REALM	已部署应用的单点登录域.
SSO_PUBLIC_KEY	单点登录域的公钥.此字段是可选的，但如果忽略，可能会使应用程序容易遭受中间人攻击。
SSO_USERNAME	访问单点登录 REST API 所需的单点登录用户. 示例值： **mySsoUser**
SSO_PASSWORD	**SSO_USERNAME** 变量定义的 Single Sign-On 用户的密码。 示例值： **6fedmL3P**
SSO_SAML_KEYSTORE	SAML 的密钥存储位置.默认为 **/etc/sso-saml-secret-volume/keystore.jks**。
SSO_SAML_KEYSTORE_PASSWORD	SAML 的密钥存储密码.默认为 **mykeystorepass**。
SSO_SAML_CERTIFICATE_NAME	用于 SAML 的密钥和证书的别名。默认为 **jboss**。
SSO_BEARER_ONLY	单点登录客户端访问类型.（可选） 示例值： **true**
SSO_CLIENT	单点登录的路径重新指向应用。默认为匹配 **module-name**。
SSO_ENABLE_CORS	如果为 **true**，请为单点登录应用程序启用跨Origin Resource Sharing (CORS)。（可选）
SSO_SECRET	用于机密访问的单点登录客户端机密。 示例值： **KZ1Qylq4**

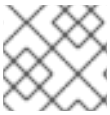
变量名称	描述
SSO_DISABLE_SSL_CERTIFICATE_VALIDATION	如果为 **true**，JBoss EAP 和红帽单点登录服务器之间的 SSL/TLS 通信不安全，例如，使用 **curl** 禁用证书验证。默认情况下不设置。 示例值： **true**

表 8.10. Secrets

变量名称	描述
SSO_SAML_KEYSTORE_SECRET	用于访问 SAML 密钥存储的机密。默认值为 **sso-app-secret**。
HTTPS_SECRET	包含密钥存储文件的 secret 名称。 示例值： **eap-ssl-secret**
SSO_TRUSTSTORE_SECRET	包含 truststore 文件的 secret 名称。用于 **sso-truststore-volume** 卷。 示例值： **sso-app-secret**

8.15. 不支持的事务恢复方案

- *OpenShift 不支持 JTS 事务。*
- *OpenShift 不支持 XTS 事务。*
- *OpenShift 不支持一些第三方用于事务完成和崩溃恢复流程的 XATerminator 接口。*
- *OpenShift 3 不支持通过 JBoss Remoting 进行事务进行事务处理。*



注意

OpenShift 4 和 EAP 操作员支持通过 JBoss 远程 传播的事务。

8.16. 包括的 JBOSS 模块

下表列出了 JBoss EAP for OpenShift 镜像中包含的 JBoss 模块。

表 8.11. 包括的 JBoss 模块

JBoss 模块
org.jboss.as.clustering.common
org.jboss.as.clustering.jgroups

JBoss 模块
org.jboss.as.ee
org.jgroups
org.openshift.ping
net.oauth.core

8.17. EAP OPERATOR: API 信息

EAP 操作器包括以下 API：

8.17.1. WildFlyServer

WildFlyServer 定义自定义 JBoss EAP 资源。

表 8.12. WildFlyServer

字段	描述	方案	必需
metadata	标准对象的元数据	ObjectMeta v1 meta	false
spec	JBoss EAP 部署的必要行为的 规范。	**WildFlyServerSpec**	true
status	最近观察 JBoss EAP 部署的状态。https://github.com/kubernetes/community/blob/master/contributors/devel/sig-architecture/api-conventions.md#spec-and-status#spec-and-status只读.	WildFlyServerStatus	false

8.17.2. WildFlyServerList

WildFlyServerList 定义 JBoss EAP 部署的列表。

表 8.13. 表

字段	描述	方案	必需
metadata	标准列表的元数据	metav1.ListMeta	false
items	**WildFlyServer**列表	**WildFlyServer**	true

8.17.3. WildFlyServerSpec

WildFlyServerSpec 是 JBoss EAP 资源所需行为的规格。

它使用一个 pod 规格的 StatefulSet，将存储指定的卷挂载到 /opt/jboss/wildfly/standalone/data 上。

表 8.14. WildFlyServerSpec

字段	描述	方案	必需
applicationImage	要部署的应用程序镜像名称	字符串	false
replicas	应用程序所需的副本数	int32]	true
standaloneConfigMap	spec 来指定如何从 **ConfigMap** 读取独立配置。	**StandaloneConfigMapSpec**	false
资源	指定 Stateful Set 的 request 或 limits 的资源 spec。**如果省略，会使用命名空间默认值。**	**Resources**	false
securityContext	**securityContext** spec，以定义 Stateful Set 创建的 pod 容器的特权和访问控制设置。如果省略，则使用默认权限。如需更多信息，请参阅 securityContext。	***corev1.SecurityContext**	false
storage	存储 spec 以指定如何使用存储。如果省略，则会使用 **EmptyDir** （在 pod 重启后不保留数据）	**StorageSpec**	false
serviceAccountName	用于运行 JBoss EAP pod 的 ServiceAccount 的名称	字符串	false
envFrom	来自 **configMap** 或 secret 的容器中存在的环境变量列表	**corev1.EnvFromSource**	false
env	容器中存在的环境变量列表	corev1.EnvVar	false

字段	描述	方案	必需
secrets	要作为容器中的卷挂载的机密名称的列表。每个 secret 都作为只读卷挂载到 **/etc/secrets/<secret name>**	字符串	false
configMaps	要作为容器中的卷挂载的 **ConfigMap** 名称列表。每个 **ConfigMap** 都被挂载为 **/etc/configmaps/<config 映射名称>**; 的只读卷	字符串	false
disableHTTPRoute	禁用创建到应用程序服务的 HTTP 端口的路由（如果省略则为false）	布尔值	false
sessionAffinity	如果每次都来自同一客户端 IP 的连接传递到同一 JBoss EAP 实例/pod（如果省略则为 false）	布尔值	false

8.17.4. 资源

资源定义 WildflyServer 资源的配置资源。如果未定义 Resources 字段或 Request 或 Limits 为空，则此资源将从 StatefulSet 中删除。此资源的描述是一个标准容器资源，并使用 corev1.ResourceRequirements 的方案。

8.17.5. StorageSpec

StorageSpec 为 WildFlyServer 资源定义配置的存储。如果未定义 EmptyDir 和 volumeClaimTemplate，则使用默认的 EmptyDir。

EAP Operator 会使用此 StorageSpec 中的信息配置 StatefulSet，以挂载专用于 JBoss EAP 使用的独立/数据目录的卷，以保留自己的数据。例如，事务日志。如果使用 EmptyDir，则 pod 重启后数据不会保留。如果 JBoss EAP 上部署的应用依赖于事务，请指定 volumeClaimTemplate，以便在 pod 重启时重复使用相同的持久性卷。

表 8.15. 表

字段	描述	方案	必需
emptyDir	**EmptyDirVolumeSource** 供 JBoss EAP **StatefulSet**使用	corev1.EmptyDirVolumeSource	false
volumeClaimTemplate	PersistentVolumeClaim spec，**用于配置** 资源要求来存储 JBoss EAP 单机数据目录。模板的名称派生自 **WildFlyServer** 名称。对应的卷以 **ReadWriteOnce** 访问模式挂载。	corev1.PersistentVolumeClaim	false

8.17.6. StandaloneConfigMapSpec

StandaloneConfigMapSpec 定义了 JBoss EAP 单机配置如何从 ConfigMap 读取。如果省略，JBoss EAP 会使用其映像中的 standalone.xml 配置。

表 8.16. StandaloneConfigMapSpec

字段	描述	方案	必需
名称	包含独立配置 XML 文件的 **ConfigMap** 名称。	字符串	true
key	**ConfigMap** 的关键，其值是独立配置 XML 文件。如果省略，spec 会找到 **standalone.xml** 键。	字符串	false

8.17.7. WildFlyServerStatus

WildFlyServerStatus 是 JBoss EAP 部署的最新观察状态。只读.

表 8.17. WildFlyServerStatus

字段	描述	方案	必需
replicas	应用程序的实际副本数	int32	true

字段	描述	方案	必需
selector	pod 的选择器(selector)供 HorizontalPodAutoscaler 使用	字符串	true
主机	主机路由到应用程序 HTTP 服务	字符串	true
pods	pod 的状态	PodStatus	true
scalingdownPods	在缩减清理过程中的 pod 数量	int32	true

8.17.8. PodStatus

PodStatus 是运行 JBoss EAP 应用程序的 Pod 的最新观察状态。

表 8.18. PodStatus

字段	描述	方案	必需
名称	pod 的名称	字符串	true
podIP	分配给 pod 的 IP 地址	字符串	true
state	缩减进程中的 pod 状态。默认情况下，其状态为 ACTIVE，这表示它将为请求提供服务。	字符串	false

更新于 2024-02-08