.NET 8.0

在 OpenShift Container Platform 中使用 .NET

在 OpenShift Container Platform 上安装并运行 .NET 8.0

Last Updated: 2024-05-15

.NET 8.0 在 OpenShift Container Platform 中使用 .NET

在 OpenShift Container Platform 上安装并运行 .NET 8.0

法律通告

摘要

本指南论述了如何在 OpenShift Container Platform 上安装并运行 .NET 8.0。

目录

使开源包含更多

红帽致力于替换我们的代码、文档和 Web 属性中有问题的语言。我们从这四个术语开始：master、slave、黑名单和白名单。由于此项工作十分艰巨，这些更改将在即将推出的几个发行版本中逐步实施。详情请查看 CTO Chris Wright 的信息。

对红帽文档提供反馈

我们感谢您对我们文档的反馈。让我们了解如何改进它。

通过 Jira 提交反馈（需要帐户）

1. 登录到 Jira 网站。
2. 在顶部导航栏中点 **Create**
3. 在 **Summary** 字段中输入描述性标题。
4. 在 **Description** 字段中输入您对改进的建议。包括文档相关部分的链接。
5. 点对话框底部的 **Create**。

第 1 章 概述

NET 镜像通过从 s2i-dotnetcore 导入镜像流定义来添加到 OpenShift 中。

镜像流定义包括 **dotnet** 镜像流，其中包含用于不同支持的 .NET 版本的 sdk 镜像。 .NET 程序的生命周期和支持政策 提供了支持版本的最新概述。

Version	Tag	Alias
.NET 6.0	dotnet:6.0-ubi8	dotnet:6.0
.NET 7.0	dotnet:7.0-ubi8	dotnet:7.0
.NET 8.0	dotnet:8.0-ubi8	dotnet:8.0

sdk 镜像具有对应的运行时镜像，它们在 **dotnet-runtime** 镜像流下定义。

容器镜像可在不同版本的 Red Hat Enterprise Linux 和 OpenShift 中工作。基于 UBI-8 的镜像(suffix -ubi8)托管在 **registry.access.redhat.com** 上，不需要身份验证。

第 2 章 安装 .NET 镜像流

要安装 .NET 镜像流，请使用带有 OpenShift Client (**oc**)二进制文件的 s2i-dotnetcore 中的镜像流定义。镜像流可以从 Linux、Mac 和 Windows 安装。

您可以在全局 **openshift** 命名空间中或本地定义 .NET 镜像流。更新 **openshift** 命名空间定义需要足够的权限。

流程

1. 安装（或更新镜像流）：

```
$ oc apply [-n namespace] -f
https://raw.githubusercontent.com/redhat-developer/s2i-
dotnetcore/main/dotnet_imagestreams.json
```

第 3 章 使用 OPENSHIFT 客户端部署应用程序

您可以使用 OpenShift Client (oc)进行应用部署。您可以从源或二进制工件部署应用程序。

3.1. 使用 oc从源部署应用程序

以下示例演示了如何使用 **oc** 部署 *example-* ***app*** 应用程序，该应用程序位于 **redhat-developer/s2i-dotnetcore-ex** GitHub 存储库的 **dotnet-8.0** 分支的 dotnet-8.0 分支中：

流程

1. 创建新的 OpenShift 项目：

```
$ oc new-project sample-project
```

2. 添加 ASP.NET Core 应用程序：

```
$ oc new-app --name=example-app 'dotnet:8.0-ubi8~https://github.com/redhat-developer/s2i-dotnetcore-ex#dotnet-8.0' --build-env DOTNET_STARTUP_PROJECT=app
```

3. 监控构建的进度：

```
$ oc logs -f bc/example-app
```

4. 构建完成后查看部署的应用程序：

```
$ oc logs -f dc/example-app
```

该应用现在可以在项目内访问。

5. **可选** ：使项目可以访问外部：

```
$ oc expose svc/example-app
```

6. 获取可共享 URL：

```
$ oc get routes
```

3.2. 使用 oc从二进制工件部署应用程序

您可以使用 .NET Source-to-Image (S2I)构建器镜像来使用您提供的二进制工件构建应用程序。

先决条件

1. 发布的应用程序。
   如需更多信息，请参阅

流程

1. 创建新的二进制构建：

```
$ oc new-build --name=my-web-app dotnet:8.0-ubi8 --binary=true
```

2. 启动构建并指定本地机器中二进制工件的路径：

```
$ oc start-build my-web-app --from-dir=bin/Release/net8.0/publish
```

3. 创建新应用程序：

```
$ oc new-app my-web-app
```

第 4 章 .NET 8.0 的环境变量

.NET 镜像支持多个环境变量来控制 .NET 应用程序的构建行为。您可以将这些变量设置为构建配置的一部分，或者将它们添加到应用源代码存储库的 **.s2i/environment** 文件中。

变量名称	描述	default
DOTNET_STARTUP_PROJECT	选择要运行的项目。这必须是项目文件（如 **csproj** 或 **fsproj**）或包含单个项目文件的文件夹。	**.**
DOTNET_ASSEMBLY_NAME	选择要运行的 assembly。这不得包含 **.dll** 扩展。把它设置为 **csproj** 中指定的输出 assembly 名称 (PropertyGroup/AssemblyName)。	**csproj** 文件的名称
DOTNET_PUBLISH_READYTORUN	当设置为 **true** 时，应用程序将提前编译。这可减少启动时间，从而减少了应用程序加载时 JIT 需要执行的工作量。	**false**
DOTNET_RESTORE_SOURCES	指定恢复操作中使用的 NuGet 软件包源的逗号分隔列表。这会覆盖 **NuGet.config** 文件中指定的所有源。此变量不能与 **DOTNET_RESTORE_CONFIGFILE** 结合使用。	
DOTNET_RESTORE_CONFIGFILE	指定用于恢复操作的 **NuGet.Config** 文件。此变量不能与 **DOTNET_RESTORE_SOURCES** 结合使用。	
DOTNET_TOOLS	指定在构建应用程序前要安装的 .NET 工具列表。可以通过使用 @<version> 来等待软件包名安装特定版本。	
DOTNET_NPM_TOOLS	指定在构建应用程序前要安装的 NPM 软件包列表。	
DOTNET_TEST_PROJECTS	指定要测试的测试项目列表。这必须是包含单个项目文件的项目文件或文件夹。为每个项目调用 **dotnet test**。	
DOTNET_CONFIGURATION	以 Debug 或 Release 模式运行应用程序。这个值应该是 **Release** 或 **Debug**。	**Release**

变量名称	描述	default
DOTNET_VERBOSITY	指定 **dotnet 构建** 命令的详细程度。设置后，环境变量会在构建开始时打印。这个变量可以被设置为 msbuild verbosity 值（**q[uiet]**、**m[inimal]**、**n[ormal]**、**d[etailed]** 和 **diag[nostic]**）。	
HTTP_PROXY, HTTPS_PROXY	配置构建和运行应用时使用的 HTTP 或 HTTPS 代理。	
DOTNET_RM_SRC	当设置为 **true** 时，镜像中不包含源代码。	
DOTNET_SSL_DIRS	**弃用**: 使用 **SSL_CERT_DIR** 替代	
SSL_CERT_DIR	指定带有要信任的额外 SSL 证书的文件夹或文件列表。证书受构建期间运行的每个进程以及构建后在镜像中运行的所有进程（包括构建的应用程序）的信任。项目可以是绝对路径（从 /开始）或源存储库中的路径（如证书）。	
NPM_MIRROR	在构建过程中使用自定义 NPM registry 镜像下载软件包。	
ASPNETCORE_URLS	这个变量设定为 **http://*:8080** 以配置 ASP.NET Core 以使用由镜像公开的端口。不建议修改它。	**http://*:8080**
DOTNET_RESTORE_DISABLE_PARALLEL	当设置为 **true** 时，会禁用并行恢复多个项目。这可减少构建容器以低 CPU 限值运行时恢复超时错误。	**false**
DOTNET_INCREMENTAL	当设置为 **true** 时，将保留 NuGet 软件包，以便将其用于增量构建。	**false**
DOTNET_PACK	当设置为 **true** 时，在 **/opt/app-root/app. tar.gz 中创建一个 tar.gz** 文件，其中包含公布的应用程序。	

第 5 章 使用 .NET 8.0 创建示例应用程序

5.1. 创建 MVC 示例应用程序

s2i-dotnetcore-ex 是 .NET 的默认 Model, View, Controller (MVC)模板应用程序。

此应用程序被 .NET S2I 镜像用作示例应用程序，并可使用 *Try Example* 链接直接从 OpenShift UI 创建。

也可以使用 OpenShift 客户端二进制文件(**oc**)创建应用。

流程

使用 **oc** 创建示例应用程序：

1. 添加 .NET 应用程序：

```
$ oc new-app dotnet:8.0-ubi8~https://github.com/redhat-developer/s2i-dotnetcore-ex#dotnet-8.0 --context-dir=app
```

2. 使应用程序可以被外部访问：

```
$ oc expose service s2i-dotnetcore-ex
```

3. 获取 sharable URL：

```
$ oc get route s2i-dotnetcore-ex
```

其他资源

- GitHub 上的 **s2i-dotnetcore-ex** 应用程序存储库

5.2. 创建 CRUD 示例应用程序

s2i-dotnetcore-persistent-ex 是一个简单 Create, Read, Update, Delete (CRUD).NET web application，它将数据存储在 PostgreSQL 数据库中。

流程

使用 **oc** 创建示例应用程序：

1. 添加数据库：

```
$ oc new-app postgresql-ephemeral
```

2. 添加 .NET 应用程序：

```
$ oc new-app dotnet:8.0-ubi8~https://github.com/redhat-developer/s2i-dotnetcore-persistent-ex#dotnet-8.0 --context-dir app
```

3. 从 **postgresql** secret 和数据库服务名称环境变量中添加环境变量：

```
$ oc set env dc/s2i-dotnetcore-persistent-ex --from=secret/postgresql -e database-service=postgresql
```

4. 使应用程序可以被外部访问：

```
$ oc expose service s2i-dotnetcore-persistent-ex
```

5. 获取 sharable URL：

```
$ oc get route s2i-dotnetcore-persistent-ex
```

其他资源

- GitHub 上的 **s2i-dotnetcore-ex** 应用程序存储库