



.NET 8.0

Getting started with .NET on OpenShift Container Platform

Installing and running .NET 8.0 on OpenShift Container Platform

.NET 8.0 Getting started with .NET on OpenShift Container Platform

Installing and running .NET 8.0 on OpenShift Container Platform

Legal Notice

Copyright © 2024 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

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

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

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

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

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

This guide describes how to install and run .NET 8.0 on OpenShift Container Platform.

Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE	3
PROVIDING FEEDBACK ON RED HAT DOCUMENTATION	4
CHAPTER 1. OVERVIEW	5
CHAPTER 2. INSTALLING .NET IMAGE STREAMS	6
CHAPTER 3. DEPLOYING APPLICATIONS WITH OPENSIFT CLIENT	7
3.1. DEPLOYING APPLICATIONS FROM SOURCE USING OC	7
3.2. DEPLOYING APPLICATIONS FROM BINARY ARTIFACTS USING OC	7
CHAPTER 4. ENVIRONMENT VARIABLES FOR .NET 8.0	9
CHAPTER 5. CREATING SAMPLE APPLICATIONS WITH .NET 8.0	12
5.1. CREATING THE MVC SAMPLE APPLICATION	12
5.2. CREATING THE CRUD SAMPLE APPLICATION	12

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see [our CTO Chris Wright's message](#).

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your feedback on our documentation. Let us know how we can improve it.

Submitting feedback through Jira (account required)

1. Log in to the [Jira](#) website.
2. Click **Create** in the top navigation bar
3. Enter a descriptive title in the **Summary** field.
4. Enter your suggestion for improvement in the **Description** field. Include links to the relevant parts of the documentation.
5. Click **Create** at the bottom of the dialogue.

CHAPTER 1. OVERVIEW

NET images are added to OpenShift by importing imagestream definitions from [s2i-dotnetcore](#).

The imagestream definitions include the **dotnet** imagestream which contains sdk images for different supported versions of .NET. [Life Cycle and Support Policies for the .NET Program](#) provides an up-to-date overview of supported versions.

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

The sdk images have corresponding runtime images which are defined under the **dotnet-runtime** imagestream.

The container images work across different versions of Red Hat Enterprise Linux and OpenShift. The UBI-8 based images (suffix -ubi8) are hosted on the [registry.access.redhat.com](#) and do not require authentication.

CHAPTER 2. INSTALLING .NET IMAGE STREAMS

To install .NET image streams, use image stream definitions from [s2i-dotnetcore](#) with the OpenShift Client (**oc**) binary. Image streams can be installed from Linux, Mac, and Windows.

You can define .NET image streams in the global **openshift** namespace or locally in a project namespace. Sufficient permissions are required to update the **openshift** namespace definitions.

Procedure

1. Install (or update) the image streams:

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

CHAPTER 3. DEPLOYING APPLICATIONS WITH OPENSIFT CLIENT

You can use OpenShift Client (`oc`) for application deployment. You can deploy applications from source or from binary artifacts.

3.1. DEPLOYING APPLICATIONS FROM SOURCE USING `oc`

The following example demonstrates how to deploy the `example-app` application using `oc`, which is in the `app` folder on the `dotnet-8.0` branch of the `redhat-developer/s2i-dotnetcore-ex` GitHub repository:

Procedure

1. Create a new OpenShift project:

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

2. Add the ASP.NET Core application:

```
$ 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. Track the progress of the build:

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

4. View the deployed application once the build is finished:

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

The application is now accessible within the project.

5. **Optional:** Make the project accessible externally:

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

6. Obtain the shareable URL:

```
$ oc get routes
```

3.2. DEPLOYING APPLICATIONS FROM BINARY ARTIFACTS USING `oc`

You can use .NET Source-to-Image (S2I) builder image to build applications using binary artifacts that you provide.

Prerequisites

1. Published application.
For more information, see

Procedure

1. Create a new binary build:

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

2. Start the build and specify the path to the binary artifacts on your local machine:

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

3. Create a new application:

```
┆ $ oc new-app my-web-app
```

CHAPTER 4. ENVIRONMENT VARIABLES FOR .NET 8.0

The .NET images support several environment variables to control the build behavior of your .NET application. You can set these variables as part of the build configuration, or add them to the `.s2i/environment` file in the application source code repository.

Variable Name	Description	Default
<code>DOTNET_STARTUP_PROJECT</code>	Selects the project to run. This must be a project file (for example, csproj or fsproj) or a folder containing a single project file.	.
<code>DOTNET_ASSEMBLY_NAME</code>	Selects the assembly to run. This must not include the .dll extension. Set this to the output assembly name specified in csproj (PropertyGroup/AssemblyName).	The name of the csproj file
<code>DOTNET_PUBLISH_READYTORUN</code>	When set to true , the application will be compiled ahead of time. This reduces startup time by reducing the amount of work the JIT needs to perform when the application is loading.	false
<code>DOTNET_RESTORE_SOURCES</code>	Specifies the space-separated list of NuGet package sources used during the restore operation. This overrides all of the sources specified in the NuGet.config file. This variable cannot be combined with DOTNET_RESTORE_CONFIGFILE .	
<code>DOTNET_RESTORE_CONFIGFILE</code>	Specifies a NuGet.Config file to be used for restore operations. This variable cannot be combined with DOTNET_RESTORE_SOURCES .	
<code>DOTNET_TOOLS</code>	Specifies a list of .NET tools to install before building the app. It is possible to install a specific version by post pending the package name with @<version> .	
<code>DOTNET_NPM_TOOLS</code>	Specifies a list of NPM packages to install before building the application.	

Variable Name	Description	Default
DOTNET_TEST_PROJECTS	Specifies the list of test projects to test. This must be project files or folders containing a single project file. dotnet test is invoked for each item.	
DOTNET_CONFIGURATION	Runs the application in Debug or Release mode. This value should be either Release or Debug .	Release
DOTNET_VERBOSITY	Specifies the verbosity of the dotnet build commands. When set, the environment variables are printed at the start of the build. This variable can be set to one of the msbuild verbosity values (q[uiet] , m[inimal] , n[ormal] , d[etailed] , and diag[nostic]).	
HTTP_PROXY, HTTPS_PROXY	Configures the HTTP or HTTPS proxy used when building and running the application, respectively.	
DOTNET_RM_SRC	When set to true , the source code will not be included in the image.	
DOTNET_SSL_DIRS	Deprecated: Use SSL_CERT_DIR instead	
SSL_CERT_DIR	Specifies a list of folders or files with additional SSL certificates to trust. The certificates are trusted by each process that runs during the build and all processes that run in the image after the build (including the application that was built). The items can be absolute paths (starting with /) or paths in the source repository (for example, certificates).	
NPM_MIRROR	Uses a custom NPM registry mirror to download packages during the build process.	

Variable Name	Description	Default
ASPNETCORE_URLS	This variable is set to http://*:8080 to configure ASP.NET Core to use the port exposed by the image. Changing this is not recommended.	http://*:8080
DOTNET_RESTORE_DISABLE_PARALLEL	When set to true , disables restoring multiple projects in parallel. This reduces restore timeout errors when the build container is running with low CPU limits.	false
DOTNET_INCREMENTAL	When set to true , the NuGet packages will be kept so they can be re-used for an incremental build.	false
DOTNET_PACK	When set to true , creates a tar.gz file at /opt/app-root/app.tar.gz that contains the published application.	

CHAPTER 5. CREATING SAMPLE APPLICATIONS WITH .NET 8.0

5.1. CREATING THE MVC SAMPLE APPLICATION

s2i-dotnetcore-ex is the default Model, View, Controller (MVC) template application for .NET.

This application is used as the example application by the .NET S2I image and can be created directly from the OpenShift UI using the *Try Example* link.

The application can also be created with the OpenShift client binary (**oc**).

Procedure

To create the sample application using **oc**:

1. Add the .NET application:

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

2. Make the application accessible externally:

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

3. Obtain the sharable URL:

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

Additional resources

- [s2i-dotnetcore-ex application repository on GitHub](#)

5.2. CREATING THE CRUD SAMPLE APPLICATION

s2i-dotnetcore-persistent-ex is a simple Create, Read, Update, Delete (CRUD) .NET web application that stores data in a PostgreSQL database.

Procedure

To create the sample application using **oc**:

1. Add the database:

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

2. Add the .NET application:

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

3. Add environment variables from the **postgresql** secret and database service name environment variable:


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

4. Make the application accessible externally:

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

5. Obtain the sharable URL:

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

Additional resources

- [s2i-dotnetcore-ex](#) application repository on GitHub