



# .NET 8.0

## OpenShift Container Platform での .NET の使用 開始

OpenShift Container Platform での .NET 8.0 のインストールと実行



# .NET 8.0 OpenShift Container Platform での .NET の使用開始

---

OpenShift Container Platform での .NET 8.0 のインストールと実行

## 法律上の通知

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<sup>®</sup> is the registered trademark of Linus Torvalds in the United States and other countries.

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

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

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

Node.js<sup>®</sup> 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<sup>®</sup> 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.

## 概要

このガイドでは、OpenShift Container Platform に .NET 8.0 をインストールして実行する方法を説明します。

---

## 目次

多様性を受け入れるオープンソースの強化 .....	3
RED HAT ドキュメントへのフィードバック (英語のみ) .....	4
第1章 概要 .....	5
第2章 .NET イメージストリームのインストール .....	6
第3章 OPENSIFT CLIENT を使用したアプリケーションのデプロイメント .....	7
3.1. OC を使用したソースからのアプリケーションのデプロイメント .....	7
3.2. OC を使用したバイナリーアーティファクトからアプリケーションのデプロイ .....	7
第4章 .NET 8.0 の環境変数 .....	9
第5章 .NET 8.0 を使用したサンプルアプリケーションの作成 .....	12
5.1. MVC サンプルアプリケーションの作成 .....	12
5.2. CRUD サンプルアプリケーションの作成 .....	12



## 多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、[Red Hat CTO である Chris Wright のメッセージ](#) をご覧ください。

## RED HAT ドキュメントへのフィードバック (英語のみ)

Red Hat ドキュメントに関するご意見やご感想をお寄せください。また、改善点があればお知らせください。

### Jira からのフィードバック送信 (アカウントが必要)

1. [Jira](#) の Web サイトにログインします。
2. 上部のナビゲーションバーで **Create** をクリックします。
3. **Summary** フィールドにわかりやすいタイトルを入力します。
4. **Description** フィールドに、ドキュメントの改善に関するご意見を記入してください。ドキュメントの該当部分へのリンクも追加してください。
5. ダイアログの下部にある **Create** をクリックします。



## 第1章 概要

NET イメージは、[s2i-dotnetcore](#) からイメージストリーム定義をインポートすることで OpenShift に追加されます。

イメージストリーム定義には、サポートされる異なるバージョンの .NET の sdk イメージが含まれる **dotnet** イメージストリームが含まれます。[.NET プログラムのライフサイクルおよびサポートポリシー](#) では、サポートされているバージョンの最新の情報をまとめています。

バージョン	タグ	エイリアス
.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 イメージには、**dotnet-runtime** イメージストリームで定義される対応するランタイムイメージがあります。

コンテナイメージは、Red Hat Enterprise Linux と OpenShift の異なるバージョン間で機能します。UBI-8 ベースのイメージ (suffix -ubi8) は [registry.access.redhat.com](#) でホストされ、認証は必要ありません。

## 第2章 .NET イメージストリームのインストール

.NET イメージストリームをインストールするには、[s2i-dotnetcore](#) のイメージストリーム定義と OpenShift Client (**oc**) バイナリーを使用してインストールされます。イメージストリームは、Linux、Mac、Windows からインストールできます。

.NET イメージストリームは、グローバルな **openshift** namespace で定義するか、プロジェクト namespace でローカルにストリームします。**openshift** namespace の定義を更新するには、十分な権限が必要です。

### 手順

1. イメージストリームをインストール (または更新) します。

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

## 第3章 OPENSIFT CLIENT を使用したアプリケーションのデプロイメント

アプリケーションのデプロイメントに OpenShift クライアント (oc) を使用できます。アプリケーションはソースまたはバイナリーアーティファクトからデプロイできます。

### 3.1. oc を使用したソースからのアプリケーションのデプロイメント

以下の例では、**oc** を使用した **example-app** アプリケーションのデプロイ方法を説明します。これは、**redhat-developer/s2i-dotnetcore-ex** GitHub リポジトリの **dotnet-8.0** ブランチの **app** ディレクトリにあります。

#### 手順

1. 新しい OpenShift プロジェクトを作成します。

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

2. ASP.NET Core アプリケーションを追加します。

```
$ 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. ビルドの進捗を追跡します。

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

4. ビルドが完了したら、デプロイされたアプリケーションを表示します。

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

これで、プロジェクト内でアプリケーションにアクセスできます。

5. オプション: プロジェクトを外部からアクセス可能にします。

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

6. 共有可能な URL を取得します。

```
$ oc get routes
```

### 3.2. oc を使用したバイナリーアーティファクトからアプリケーションのデプロイ

.NET Source-to-Image (S2I) ビルダージェイメージを使用して、提供するバイナリーアーティファクトを使用してアプリケーションをビルドできます。

#### 前提条件

1. 公開済みアプリケーション。

詳細は以下を参照してください。

## 手順

1. 新しいバイナリービルドを作成します。

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

2. ビルドを開始し、ローカルマシンのバイナリーアーティファクトへのパスを指定します。

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

3. 新規アプリケーションを作成します。

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

## 第4章 .NET 8.0 の環境変数

.NET イメージは、.NET アプリケーションのビルド動作を制御する複数の環境変数をサポートします。これらの変数はビルド設定の一部として設定したり、アプリケーションのソースコードリポジトリの `.s2i/environment` ファイルに追加できます。

変数名	説明	デフォルト
<code>DOTNET_STARTUP_PROJECT</code>	実行するプロジェクトを選択します。これは、プロジェクトファイル ( <b>csproj</b> 、 <b>fsproj</b> など) またはプロジェクトファイルを1つ含むディレクトリである必要があります。	.
<code>DOTNET_ASSEMBLY_NAME</code>	実行するアセンブリを選択します。これには <code>.dll</code> 拡張子を含めないでください。これを、 <b>csproj</b> で指定した出力アセンブリ名 (PropertyGroup/AssemblyName) に設定します。	<b>csproj</b> ファイルの名前
<code>DOTNET_PUBLISH_READYTORUN</code>	<b>true</b> に設定すると、アプリケーションは事前にコンパイルされます。これにより、アプリケーションの読み込み時に JIT が必要な作業量が削減されるため、起動時間が短縮されます。	<b>false</b>
<code>DOTNET_RESTORE_SOURCES</code>	復元操作中使用される NuGet パッケージソースのスペース区切りリストを指定します。これにより、 <b>NuGet.config</b> ファイルで指定されたすべてのソースが上書きされます。この変数を <b>DOTNET_RESTORE_CONFIGFILE</b> と組み合わせることはできません。	
<code>DOTNET_RESTORE_CONFIGFILE</code>	復元操作中使用される <b>NuGet.Config</b> ファイルを指定します。この変数を <b>DOTNET_RESTORE_SOURCES</b> と組み合わせることはできません。	
<code>DOTNET_TOOLS</code>	アプリをビルドする前にインストールする .NET ツールのリストを指定します。@<version> でパッケージ名を保留することにより、特定のバージョンをインストールできます。	

変数名	説明	デフォルト
DOTNET_NPM_TOOLS	アプリケーションをビルドする前にインストールする NPM パッケージのリストを指定します。	
DOTNET_TEST_PROJECTS	テストするテストプロジェクトのリストを指定します。これは、プロジェクトファイルまたは、単一のプロジェクトファイルを含むディレクトリである必要があります。各項目に対して <b>dotnet test</b> が呼び出されます。	
DOTNET_CONFIGURATION	Debug モードまたは Release モードでアプリケーションを実行します。この値は、 <b>Release</b> または <b>Debug</b> でなければなりません。	<b>Release</b>
DOTNET_VERBOSITY	<b>dotnet build</b> コマンドの詳細度を指定します。設定すると、環境変数はビルドの開始時に出力されます。この変数は、msbuild の詳細度 ( <b>q[uiet]</b> 、 <b>m[inimal]</b> 、 <b>n[ormal]</b> 、 <b>d[etailed]</b> 、および <b>diag[nostic]</b> ) のいずれかに設定できます。	
HTTP_PROXY, HTTPS_PROXY	アプリケーションをビルドおよび実行するときにそれぞれ使用される HTTP または HTTPS プロキシを設定します。	
DOTNET_RM_SRC	<b>true</b> に設定すると、ソースコードはイメージに含まれません。	
DOTNET_SSL_DIRS	<b>非推奨</b> : 代わりに <b>SSL_CERT_DIR</b> を使用してください	
SSL_CERT_DIR	信頼する追加の SSL 証明書を含むディレクトリまたはファイルのリストを指定します。証明書は、ビルド中に実行する各プロセスと、ビルド後のイメージで実行するすべてのプロセス (ビルドされたアプリケーションを含む) により信頼されます。項目は、絶対パス (/ で始まる) またはソースリポジトリのパス (証明書など) にすることができます。	

変数名	説明	デフォルト
NPM_MIRROR	ビルドプロセス中にカスタム NPM レジストリーミラーを使用してパッケージをダウンロードします。	
ASPNETCORE_URLS	この変数は <b>http://*:8080</b> に設定され、イメージにより公開されるポートを使用するように ASP.NET Core を設定します。これを変更することは推奨されません。	<b>http://*:8080</b>
DOTNET_RESTORE_DISABLE_PARALLEL	<b>true</b> に設定すると、複数のプロジェクトを並行して復元できなくなります。これにより、CPU 制限の値が低く設定された状態で、ビルドコンテナが実行されている場合にも復元タイムアウトのエラーが減少します。	<b>false</b>
DOTNET_INCREMENTAL	<b>true</b> に設定すると、NuGet パッケージは保持され、インクリメンタルビルドに再利用できます。	<b>false</b>
DOTNET_PACK	<b>true</b> に設定すると、公開アプリケーションを含む <b>tar.gz</b> ファイルが <b>/opt/app-root/app.tar.gz</b> に作成されます。	

## 第5章 .NET 8.0 を使用したサンプルアプリケーションの作成

### 5.1. MVC サンプルアプリケーションの作成

**s2i-dotnetcore-ex** は、.NET のデフォルトの .NET Core Model、View、Controller (MVC) テンプレートアプリケーションです。

このアプリケーションは、.NET S2I イメージによってサンプルアプリケーションとして使用され、**Try Example** リンクを使用して OpenShift UI から直接作成できます。

アプリケーションは、OpenShift クライアントバイナリー (**oc**) を使用して作成することもできます。

#### 手順

**oc** を使用してサンプルアプリケーションを作成するには、以下を行います。

1. .NET アプリケーションを追加します。

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

2. アプリケーションによる外部アクセスを可能にします。

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

3. 共有可能な URL を取得します。

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

#### 関連情報

- [GitHub の s2i-dotnetcore-ex アプリケーションリポジトリ](#)

### 5.2. CRUD サンプルアプリケーションの作成

**s2i-dotnetcore-persistent-ex** は、PostgreSQL データベースにデータを格納する単純な Create、Read、Update、Delete (CRUD) の .NET Core Web アプリケーションです。

#### 手順

**oc** を使用してサンプルアプリケーションを作成するには、以下を行います。

1. データベースを追加します。

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

2. .NET アプリケーションを追加します。

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

3. **postgresql** シークレットおよびデータベースサービス名環境変数から環境変数を追加します。



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

4. アプリケーションによる外部アクセスを可能にします。

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

5. 共有可能な URL を取得します。

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

## 関連情報

- [GitHub の \*\*s2i-dotnetcore-ex\*\* アプリケーションリポジトリ](#)