.NET 9.0

Getting started with .NET 9.0 on RHEL 8

Installing and running .NET 9.0 on RHEL 8

Last Updated: 2025-09-15

.NET 9.0 Getting started with .NET 9.0 on RHEL 8

Installing and running .NET 9.0 on RHEL 8

Legal Notice

Abstract

This guide describes how to install and run .NET 9.0 on RHEL 8.

Table of Contents

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. INTRODUCING .NET 9.0

.NET is a general-purpose development platform featuring automatic memory management and modern programming languages. Using .NET, you can build high-quality applications efficiently. .NET is available on Red Hat Enterprise Linux (RHEL) and OpenShift Container Platform through certified containers.

.NET offers the following features:

- The ability to follow a microservices-based approach, where some components are built with .NET and others with Java, but all can run on a common, supported platform on RHEL and OpenShift Container Platform.
- The capacity to more easily develop new .NET workloads on Microsoft Windows. You can deploy and run your applications on either RHEL or Windows Server.
- A heterogeneous data center, where the underlying infrastructure is capable of running .NET applications without having to rely solely on Windows Server.

CHAPTER 2. INSTALLING .NET 9.0

.NET 9.0 is included in the AppStream repositories for RHEL 8. The AppStream repositories are enabled by default on RHEL 8 systems.

You can install the .NET 9.0 runtime with the latest 9.0 Software Development Kit (SDK). When a newer SDK becomes available for .NET 9.0, you can install it by running **sudo dnf install**.

Prerequisites

- Installed and registered RHEL 8.10 with attached subscriptions.
  For more information, see Interactively installing RHEL from installation media .

Procedure

- Install .NET 9.0 and all of its dependencies:

```
$ sudo dnf install dotnet-sdk-9.0 -y
```

Verification

- Verify the installation:

```
$ dotnet --info
```

  The output returns the relevant information about the .NET installation and the environment.

CHAPTER 3. CREATING AN APPLICATION USING .NET 9.0

Learn how to create a C# "Hello World" application.

Procedure

1. Create a new Console application in a directory called **my-app**:

```
$ dotnet new console --output my-app
```

2. Run the project:

```
$ dotnet run --project my-app
```

The output returns:

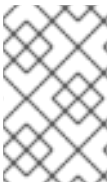
```
Hello World!
```

CHAPTER 4. PUBLISHING APPLICATIONS WITH .NET 9.0

.NET 9.0 applications can be published to use a shared system-wide version of .NET or to include .NET.

The following methods exist for publishing .NET 9.0 applications:

- Self-contained deployment (SCD) - The application includes .NET. This method uses a runtime built by Microsoft.
- Framework-dependent deployment (FDD) - The application uses a shared system-wide version of .NET.



NOTE

When publishing an application for RHEL, Red Hat recommends using FDD, because it ensures that the application is using an up-to-date version of .NET, built by Red Hat, that uses a set of native dependencies.

Prerequisites

- Existing .NET application.
  For more information about how to create a .NET application, see Creating an application using .NET.

4.1. PUBLISHING .NET APPLICATIONS

The following procedure outlines how to publish a framework-dependent application.

Procedure

1. Publish the framework-dependent application:

   ```
   $ dotnet publish my-app -f net9.0
   ```

   Replace *my-app* with the name of the application you want to publish.

2. Optional: If the application is for RHEL only, trim out the dependencies needed for other platforms:

   ```
   $ dotnet publish my-app -f net9.0 -r rhel.8-architecture --self-contained false
   ```

   - Replace *architecture* based on the platform you are using:
     - For Intel: **x64**
     - For IBM Z and LinuxONE: **s390x**
     - For 64-bit Arm: **arm64**
     - For IBM Power: **ppc64le**

CHAPTER 5. RUNNING .NET 9.0 APPLICATIONS IN CONTAINERS

Use the **ubi8/dotnet-90-runtime** image to run a .NET application inside a Linux container.

The following example uses Podman.

Procedure

1. Create a new MVC project in a directory called **mvc_runtime_example**:

```
$ dotnet new mvc --output mvc_runtime_example
```

2. Publish the project:

```
$ dotnet publish mvc_runtime_example -f net9.0 /p:PublishProfile=DefaultContainer
/p:ContainerBaseImage=registry.access.redhat.com/ubi8/dotnet-90-runtime:latest
```

3. Run your image:

```
$ podman run -rm -p8080:8080 mvc_runtime_example
```

4. View the application running in the container:

```
$ xdg-open http://127.0.0.1:8080
```

CHAPTER 6. MIGRATION FROM PREVIOUS VERSIONS OF .NET

6.1. MIGRATION FROM PREVIOUS VERSIONS OF .NET

Microsoft provides instructions for migrating from most previous versions of .NET Core.

If you are using a version of .NET that is no longer supported or want to migrate to a newer .NET version to expand functionality, see the following articles:

- Migrate from ASP.NET Core 8.0 to 9.0
- Migrate from ASP.NET Core 7.0 to 8.0
- Migrate from ASP.NET Core 6.0 to 7.0
- Migrate from ASP.NET Core 5.0 to 6.0
- Migrate from ASP.NET Core 3.1 to 5.0
- Migrate from ASP.NET Core 3.0 to 3.1
- Migrate from ASP.NET Core 2.2 to 3.0
- Migrate from ASP.NET Core 2.1 to 2.2
- Migrate from .NET Core 2.0 to 2.1
- Migrate from ASP.NET to ASP.NET Core
- Migrating .NET Core projects from project.json
- Migrate from project.json to .csproj format

NOTE

If migrating from .NET Core 1.x to 2.0, see the first few related sections in Migrate from ASP.NET Core 1.x to 2.0. These sections provide guidance that is appropriate for a .NET Core 1.x to 2.0 migration path.

6.2. PORTING FROM .NET FRAMEWORK

Refer to the following Microsoft articles when migrating from .NET Framework:

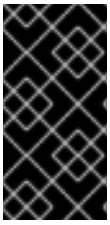
- For general guidelines, see Porting to .NET Core from .NET Framework .
- For porting libraries, see Porting to .NET Core - Libraries .
- For migrating to ASP.NET Core, see Migrating to ASP.NET Core.

Several technologies and APIs present in the .NET Framework are not available in .NET Core and .NET. If your application or library requires these APIs, consider finding alternatives or continue using the .NET Framework. .NET Core and .NET do not support the following technologies and APIs:

- Desktop applications, for example, Windows Forms and Windows Presentation Foundation (WPF)
- Windows Communication Foundation (WCF) servers (WCF clients are supported)
- .NET remoting

Additionally, several .NET APIs can only be used in Microsoft Windows environments. The following list shows examples of these Windows-specific APIs:

- **Microsoft.Win32.Registry**
- **System.AppDomains**
- **System.Security.Principal.Windows**



IMPORTANT

Several APIs that are not supported in the default version of .NET may be available from the Microsoft.Windows.Compatibility NuGet package. Be careful when using this NuGet package. Some of the APIs provided (such as **Microsoft.Win32.Registry**) only work on Windows, making your application incompatible with Red Hat Enterprise Linux.