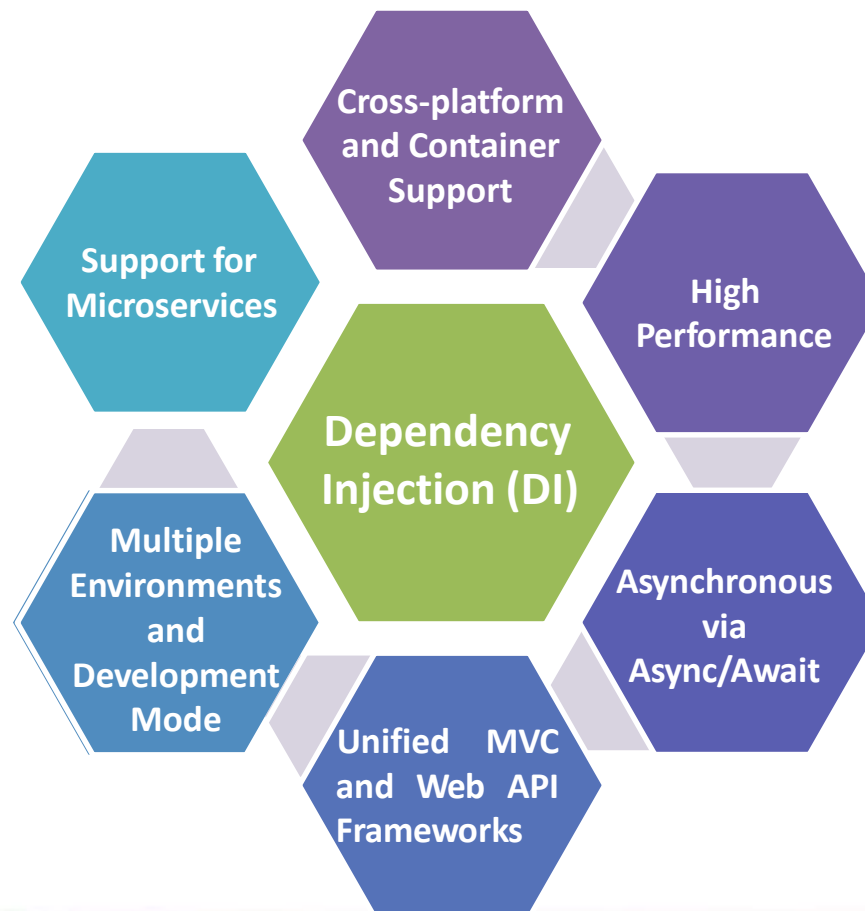


Session: **18**

.NET Core Development

- ◆ Explain what is .NET Core
- ◆ List the differences between .NET Core and .NET Framework
- ◆ Explain how to run a program on .NET Core

- .NET Core is an open source .NET Framework platform available for Windows, Linux, and Mac OS.
- Following are the key features of .NET Core:



Where and When to Use .NET Core

Where

- Docker containers are being employed.
- Server apps can be set up cross-platform to Docker containers.
- For containers.

When

- There is a need for a high performance and scalable system.
- Users are executing several .NET versions side-by-side.
- To set up applications that have dependencies on diverse versions of frameworks.
- Command Line Interface (CLI) control is required.

- Following are some of the features that are currently not supported in .NET Core and hence, not recommended in situations that need these:

Certain.NET
features and
libraries and
extensions

Windows Forms
and WPF
applications

Third-party
library support

ASP.NET Web
Forms

Partial support
for VB.NET and F#

Windows
Communication
Foundation
(WCF)

SignalR

.NET Framework in Comparison with .NET Core(1-2)

As .NET Framework can also work with Docker and Windows Containers, utilizing it in the following situations is possible:



When it is already being employed.

When utilizing third-party libraries or NuGet packages that are not supported in .NET Core.

When utilizing technologies that are not yet offered by .NET Core.

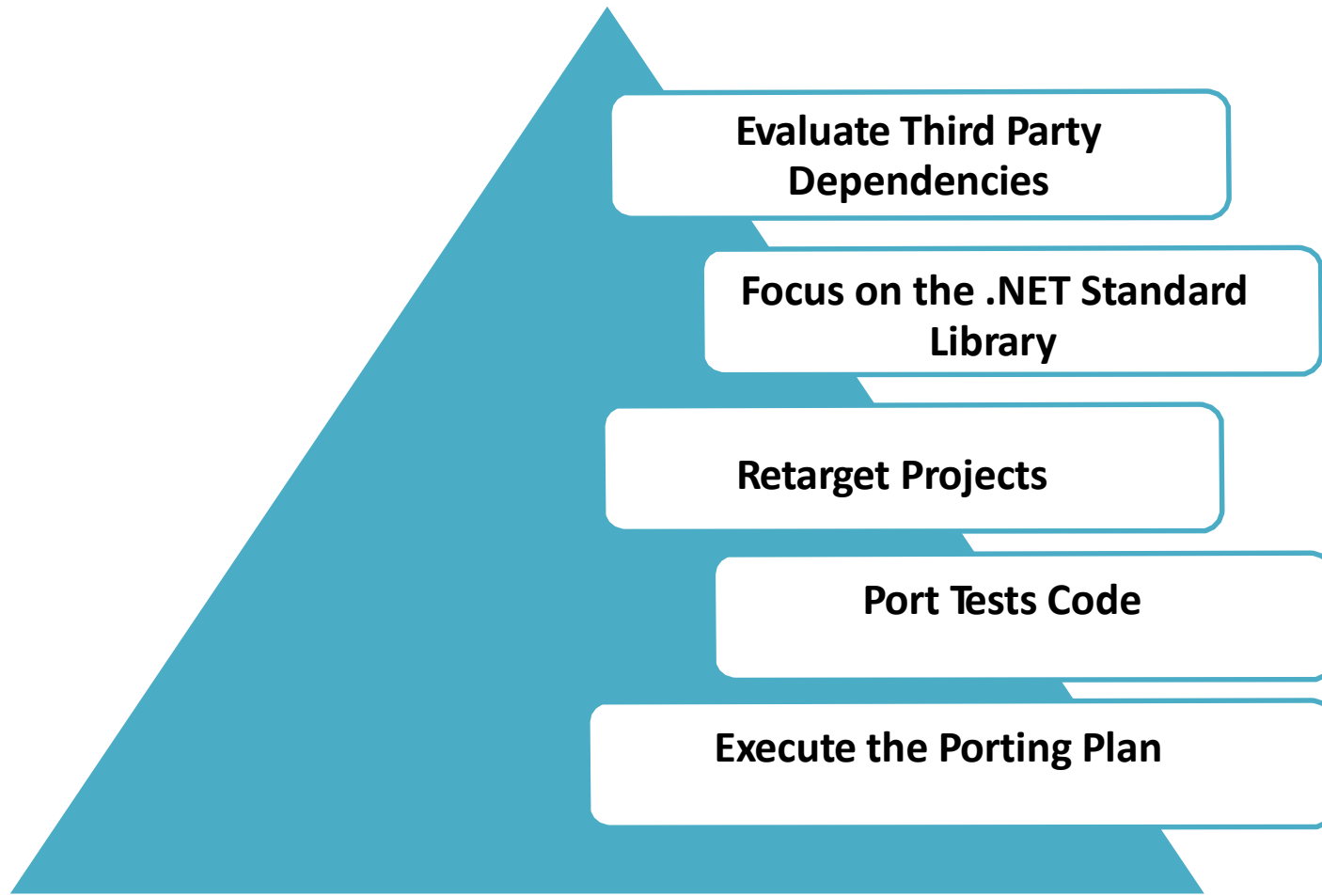
When .NET Core is not supported by the platform.

Following are the situations in which .NET Framework should not be used:

- ◆ When several OS platforms are needed.
- ◆ When high performance and scalability are required.
- ◆ When .NET Core works.
- ◆ When open source framework is needed.



Porting from .NET Framework to .NET Core



Differences Between .NET Framework and .NET Core (1-3)

.NET Framework	.NET Core
It was made available as a licensed and proprietary software framework.	It was released as an open source software framework.
It allows users to create applications for a single platform - Windows.	It accommodates three different OSes - Windows, OS X, and Linux.
It must be set up as a single package and runtime environment for Windows.	It can be packaged and set up irrespective of the underlying OS.
It uses CLR for managing libraries and compilation purposes.	It utilizes a redesigned CLR known as CoreCLR and a modular collection of libraries known as CoreFX.
If utilizing Web applications, users can employ a robust Web application framework, such as ASP.NET.	It has a redesigned version of ASP.NET.

Differences between .NET Framework and .NET Core (2-3)

.NET Framework	.NET Core
Users must deploy Web applications only on Internet Information Server.	The Web applications can be run in several ways.
It has extensive APIs for creating cloud-based applications.	It has features that ease the creation and deployment of Cloud-based application.
It does not have any robust framework or tools to simplify mobile app development.	It is compatible with Xamarin via the .NET Standard Library.
It provides few options for developing microservice oriented systems	It allows the creation of microservice oriented systems rapidly.
It requires additional infrastructure to achieve scalability.	It enhances the performance and scalability without having to deploy extra hardware or infrastructure.

Differences between .NET Framework and .NET Core (3-3)

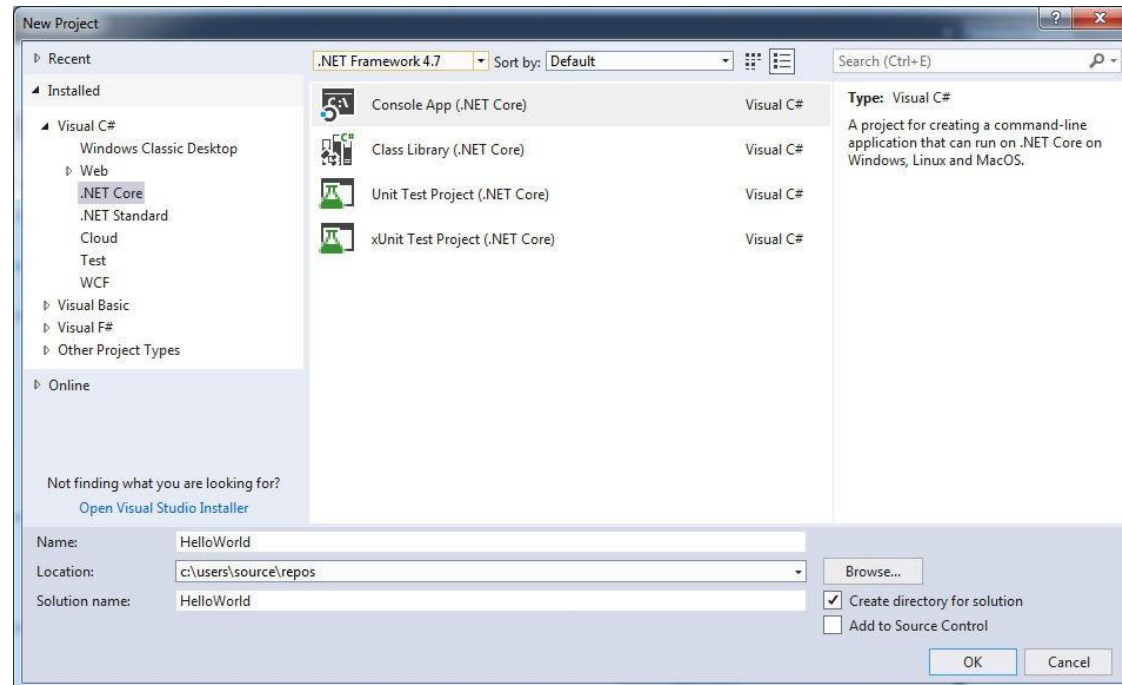
Following table summarizes which of the two (.NET Framework or .NET Core) is preferred:

Feature	.NET Framework	.NET Core
High-performance and scalable system without UI		✓
Docker containers support	✓	✓
Heavily rely on command line		✓
Cross-platform needs		✓
Using microservices	✓	✓
UI centric Web applications	✓	
Windows client applications using Windows Forms and WPF	✓	
Already have a pre-configured environment and systems	✓	
Stable version for immediate need to build and deploy	✓	
Has experienced .NET team		✓
Time is not an issue, experiments are acceptable and no rush to deployment		✓

Running a Program on .NET Core (1-2)

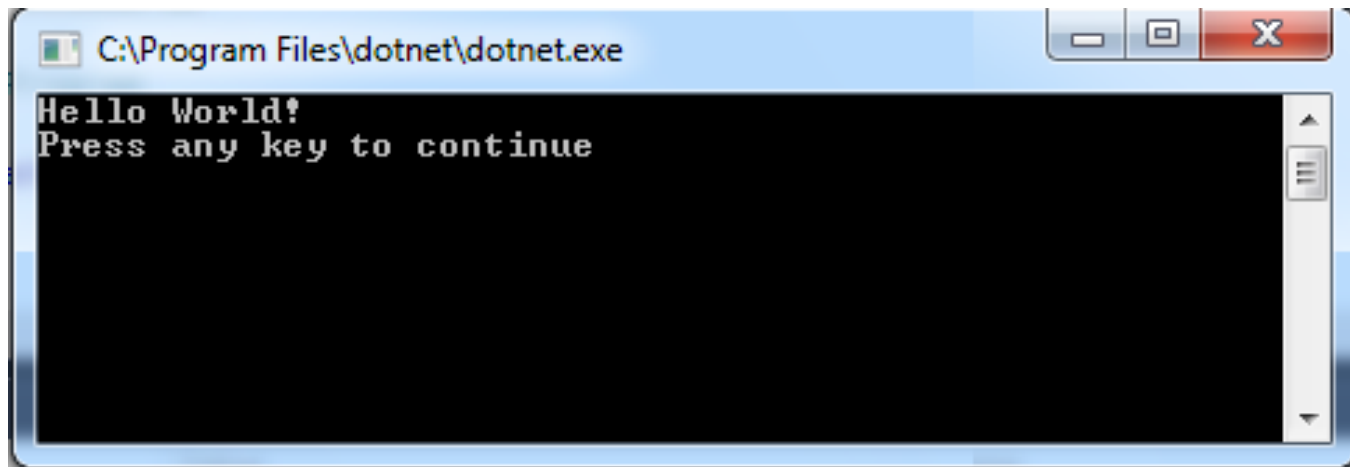
Following are the steps to create a simple 'Hello World' console application:

1. Start Visual Studio 2017. Click File→New→Project. The New Project dialog appears as shown in the figure.
 - a. Choose Visual C# followed by the .NET Core node.
 - b. Next, choose the Console App (.NET Core) project template.
 - c. Provide the name 'HelloWorld' in the Name field and click OK.



New Project Dialog Box

2. On the menu bar, click Build → Build Solution.
3. Execute the program. Following figure shows the output of the program.

A screenshot of a Windows command prompt window. The title bar shows the file path "C:\Program Files\dotnet\dotnet.exe". The window has standard Windows window controls (minimize, maximize, close) on the right. The main area is black with white text. The text displayed is "Hello World!" on the first line and "Press any key to continue" on the second line. A vertical scrollbar is visible on the right side of the text area.

```
C:\Program Files\dotnet\dotnet.exe
Hello World!
Press any key to continue
```

Output of the Program

- ◆ .NET Core is an open source .NET Framework platform available for Windows, Linux, and Mac OS.
- ◆ The key features of .NET Core are Cross-platform and container support, High performance, Asynchronous via async/await, Unified MVC and Web API Frameworks, Multiple environments and development mode, Dependency injection and Support for Microservices.
- ◆ .NET Core is a perfect fit if there is a need for a high performance and scalable system.
- ◆ Microsoft suggests executing .NET Core along with ASP.NET Core to obtain best performance and scale.
- ◆ .NET Core is usable if users are executing several .NET versions side-by-side.
- ◆ Though there are a lot of instances when .NET Core can be utilized, there are situations in which it is not recommended for use.