



Overview of the new .NET Core and .NET Platform Standard

Alex Thissen
Lead Consultant at Xpirit
@alexthissen



DOT NEXT

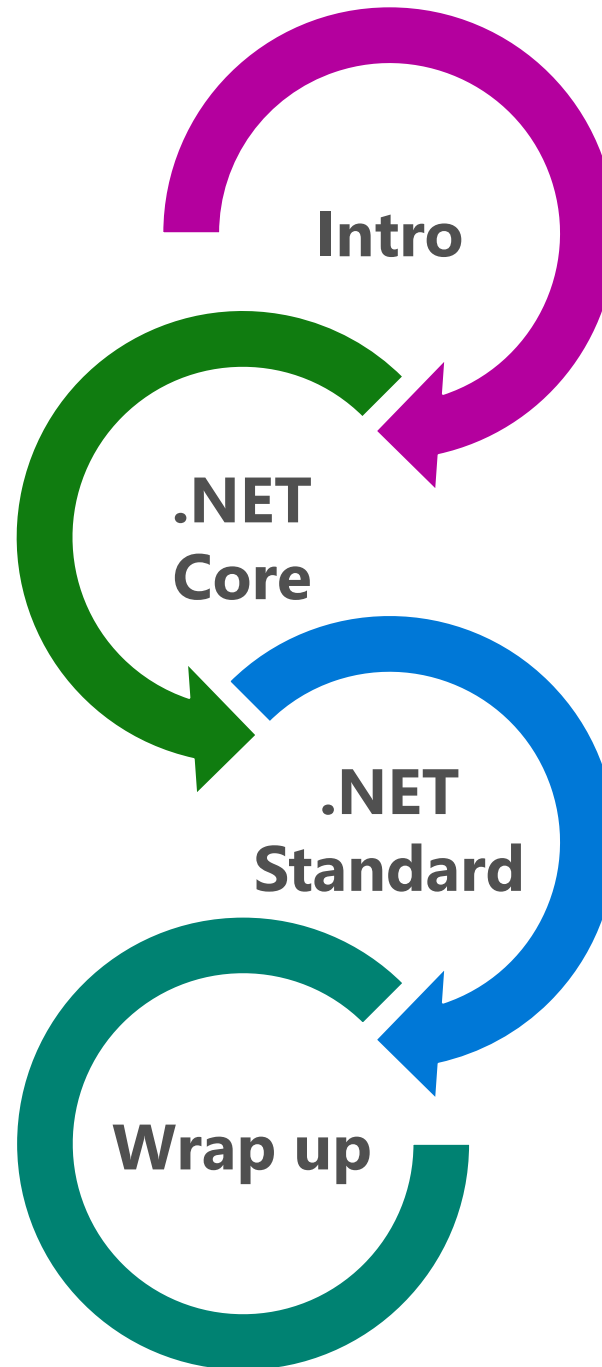


Agenda

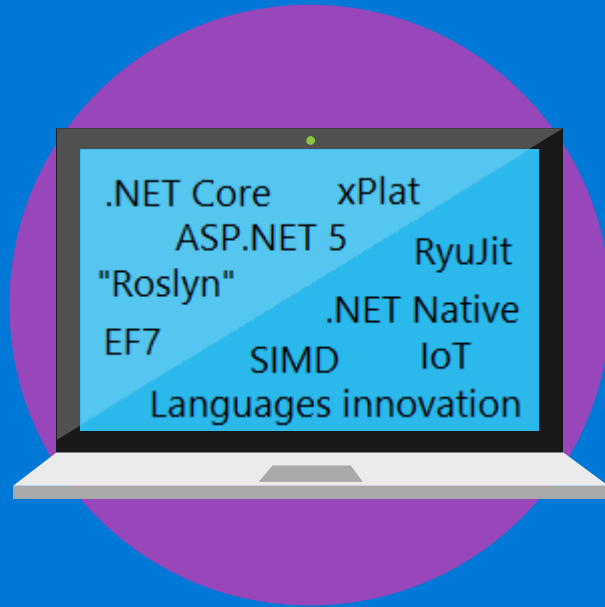
DOT
NEXT

- Runtimes
- Frameworks
- Tooling
- Compilers
- Deployment

- Summary
- Questions and answers



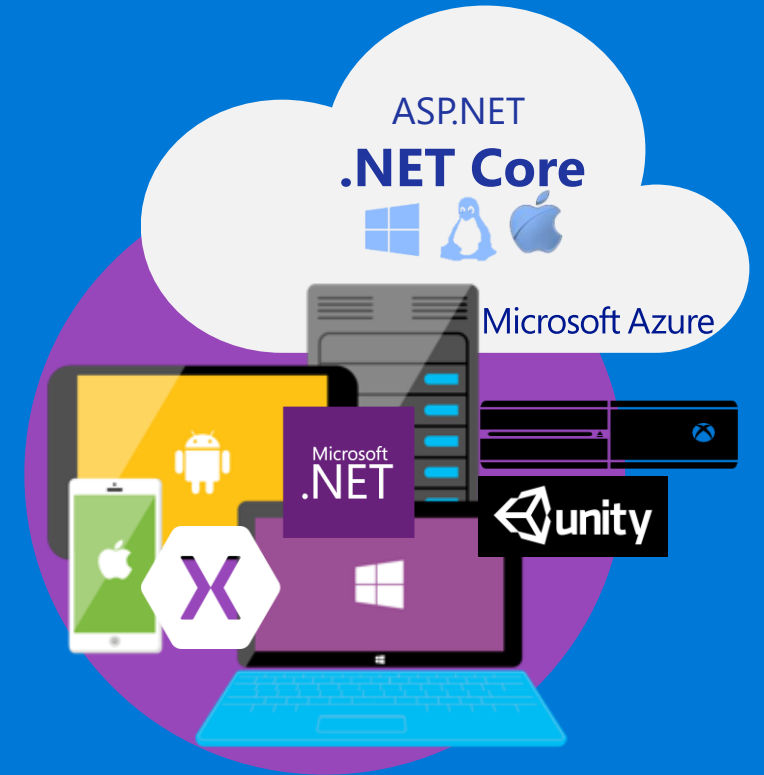
- Understanding .NET Standard
- Versioning



Innovation



Openness

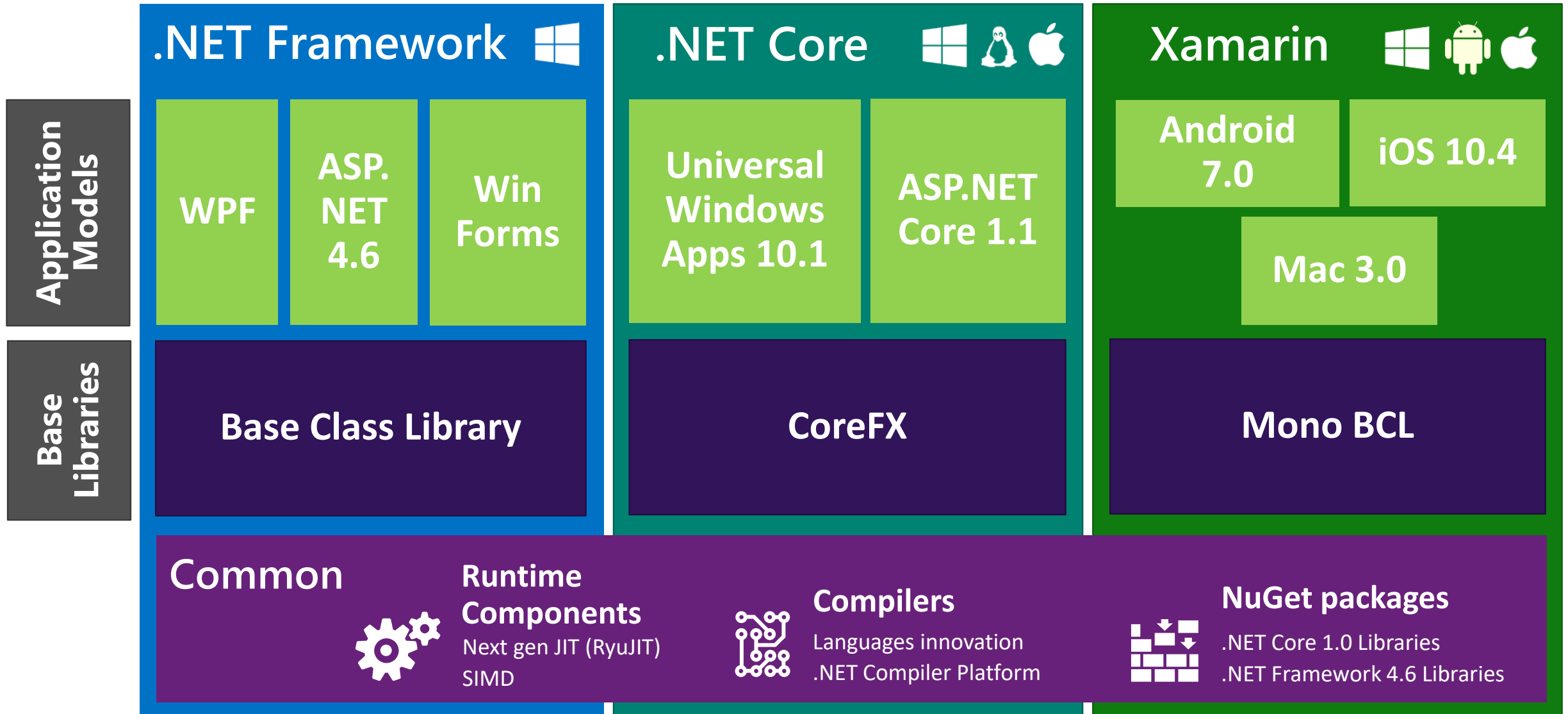


Any app, any
platform

.NET Core
to the rescue

Re-introducing .NET Core (again)

The big picture of .NET Platforms



.NET Core

Runtimes

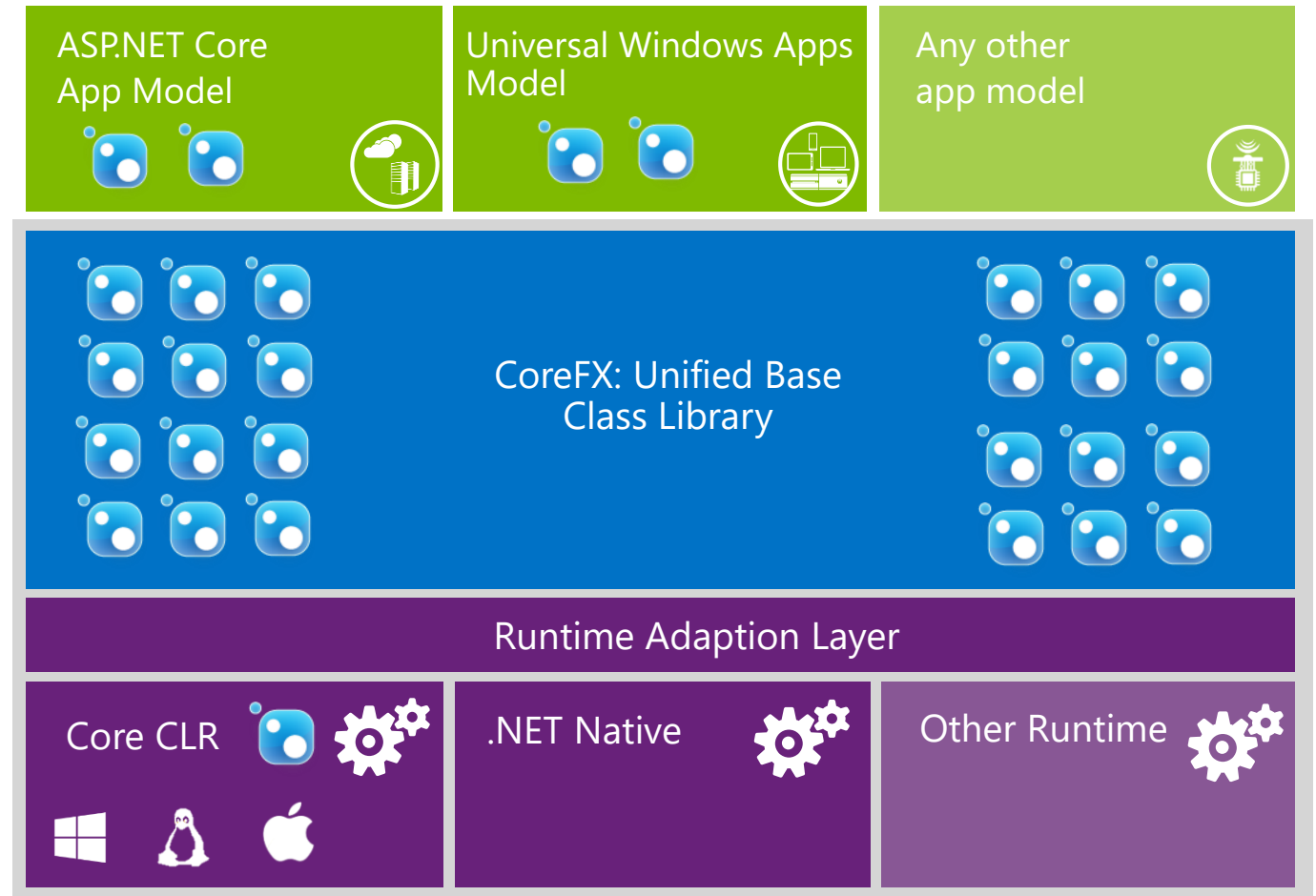
- Cross-platform implementations of CoreCLR
- CoreRT for AOT scenarios

CoreFX libraries

- Standards based BCL
- Platform and OS abstractions
- Delivered in NuGet packages
- Shared Framework

SDK

- Command-line tools
- Side by side installation
- C# and F#
- Driver dotnet.exe



Demo

Investigating Hello World
in .NET Core

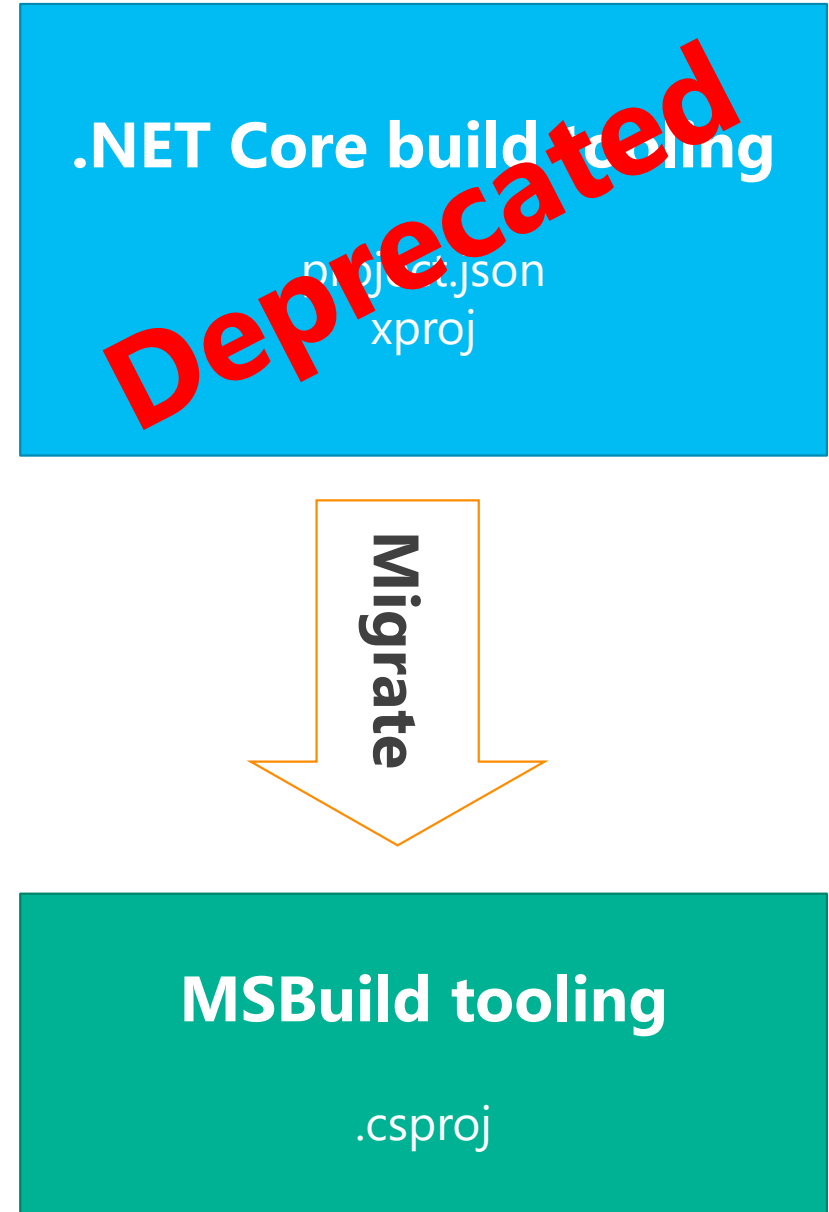
Return of MSBuild

Replacing project.json

- Unified build story across projects and platforms
- Migration with **dotnet migrate**

New and improved

- Simplified project files
 - Readable (no GUIDs)
 - Extensions to original XML
- Cross-platform implementation of msbuild.exe



Packages and more packages

NuGet Packages

Building blocks of .NET Core

Metapackages

Describes a set of packages meaningful and tested together

Easier than referencing individual packages

NETStandard.Library

Microsoft.NETCore.App

Microsoft.NETCore.Portable.Compatibility

.NET Core packages

- ▶  Microsoft.Win32.Primitives (4.0.1)
- ▶  System.AppContext (4.1.0)
- ▶  System.Collections (4.0.11)
- ▶  System.Collections.Concurrent (4.0.12)
- ▶  System.Console (4.0.0)
- ▶  System.Diagnostics.Debug (4.0.11)
- ▶  System.Diagnostics.Tools (4.0.1)
- ▶  System.Diagnostics.Tracing (4.1.0)
- ▶  System.Globalization (4.0.11)
- ▶  System.Globalization.Calendars (4.0.1)
- ▶  System.IO (4.1.0)
- ▶  System.IO.Compression (4.1.0)
- ▶  System.IO.Compression.ZipFile (4.0.1)
- ▶  System.IO.FileSystem (4.0.1)
- ▶  System.IO.FileSystem.Primitives (4.0.1)
- ▶  System.Linq (4.1.0)

Multi-targeting and multiple runtimes

Compile for targets frameworks

.NET Core applications	netcoreapp1.0
.NET Standard Libraries	netstandard1.6
.NET Full Framework	net46

More [frameworks](#) and [TFMs](#)

Runtimes for supported Operating Systems

Growing list

ARM support in works

Identified by RID

Linux

- Red Hat Enterprise Linux 7.2
- CentOS 7.1+
- Debian 8.2+
- Fedora 23
- Linux Mint 17.1, 18
- OpenSUSE 13.2, 42.1
- Oracle Linux 7.1
- Ubuntu 14.04 & 16.04

Mac OS X 10.11, 10.12

Windows

- Windows 7+ / Server 2012 R2+
- Windows Nano Server TP5
- Windows Server Core
- Windows Server 2016

Runtime Identifiers

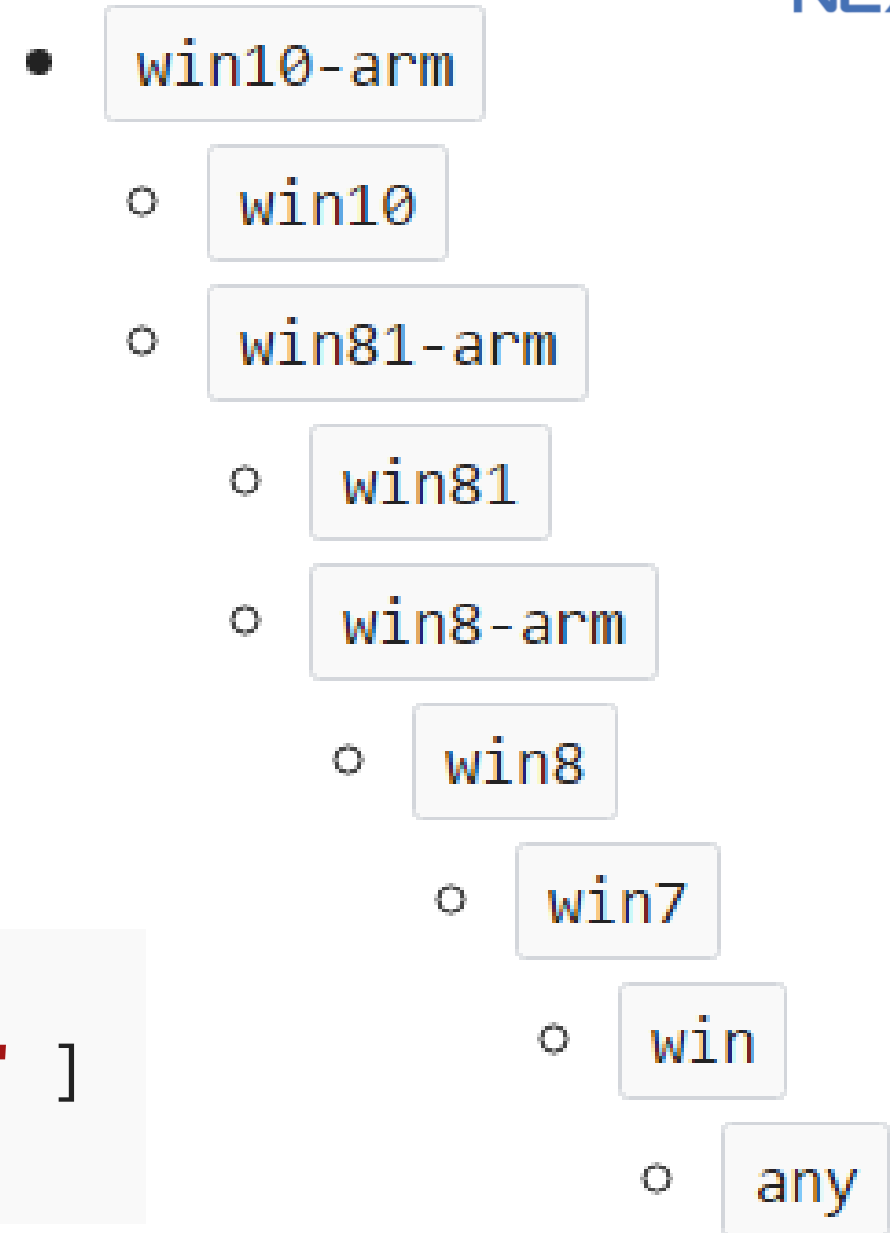
Unique identifiers for OS-specific runtime

Format: [os].[version]-[arch]

Defined in *runtime.json* of Microsoft.NETCore.Platforms package

#imports similar to **imports** in project.json dependencies

```
"osx.10.11-x64": {  
  "#import": [ "osx.10.11", "osx.10.10-x64" ]  
}
```

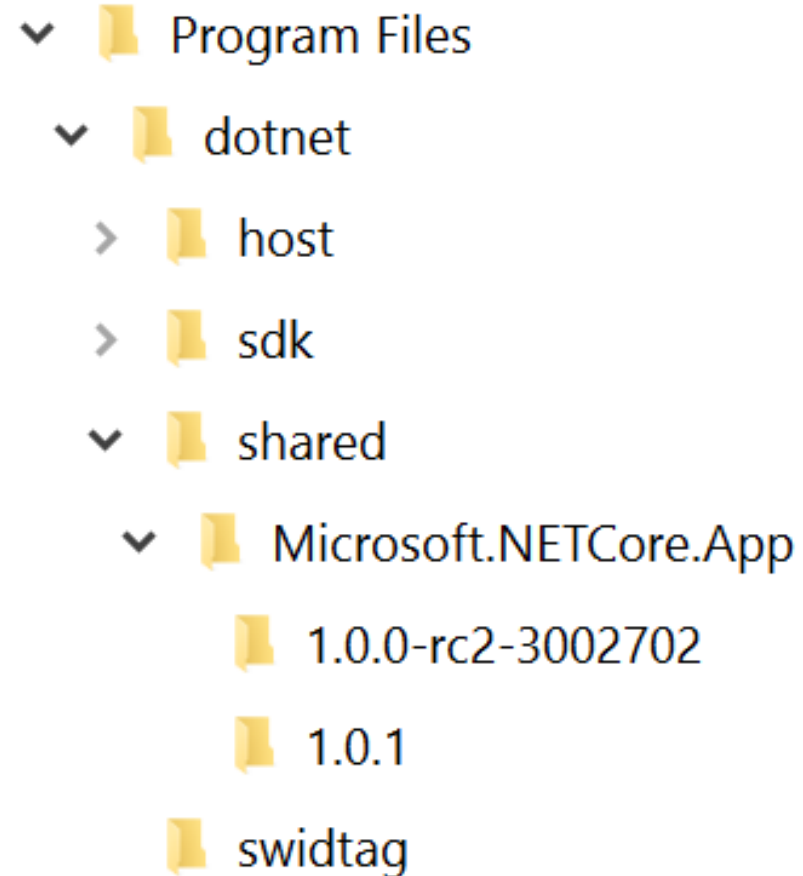


.NET Core application deployment

Framework dependent

- Relies on shared system-wide version of .NET Core: "Shared Framework"
- Contains only own code and dependencies
- Launched by **dotnet.exe** driver
- Small package size

Self-contained



still

re

specific

Command-line tooling

Driver dotnet.exe

1. Runs portable applications
2. Executes verbs

Replaces DNX and DNVM tools

Extensibility model for running new verbs

```
<ItemGroup>  
  <DotNetCliToolReference Include="dotnet-sdklist">  
    <Version>0.1.0-*</Version>  
  </DotNetCliToolReference>  
</ItemGroup>
```

Demo

Publishing and packaging
.NET Core apps

Runtime-specific self-contained deployments

Windows 7+

-  clrcompression.dll
-  clretwrc.dll
-  clrjit.dll
-  coreclr.dll
-  dbgshim.dll

-  dotnet-sdklist.deps.json
-  dotnet-sdklist.dll
-  dotnet-sdklist.exe
-  dotnet-sdklist.pdb
-  dotnet-sdklist.runtimeconfig.json

-  ext-ms-win-advapi32-encryptedfile-l1-1-...
-  ext-ms-win-ntuser-keyboard-l1-2-1.dll

-  hostfxr.dll
-  hostpolicy.dll

OSX 10.11

-  dotnet-sdklist
-  dotnet-sdklist.deps.json
-  dotnet-sdklist.dll
-  dotnet-sdklist.pdb
-  dotnet-sdklist.runtimeconfig.json

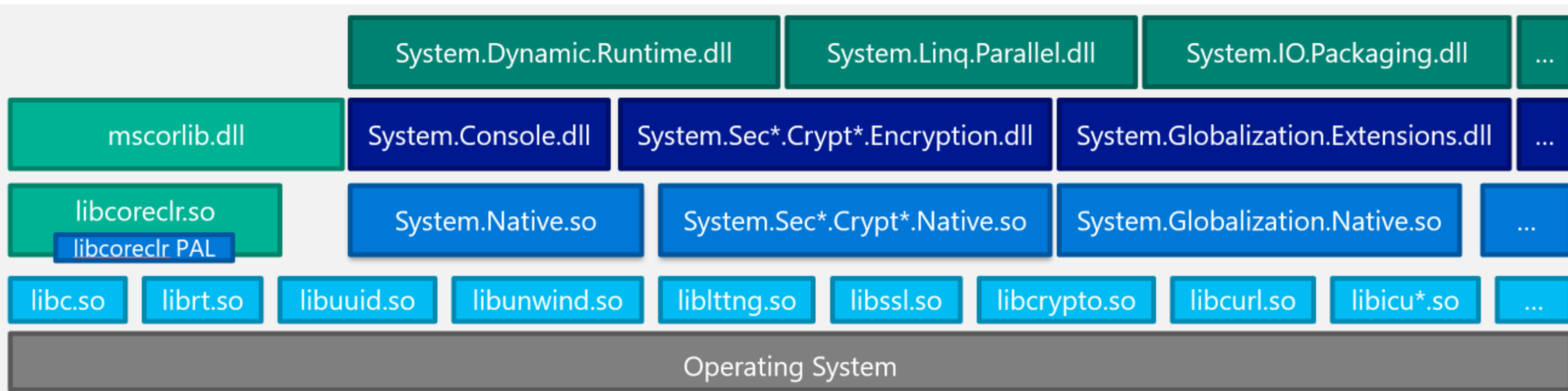
-  libclrjit.dylib
-  libcoreclr.dylib
-  libdbgshim.dylib
-  libhostfxr.dylib
-  libhostpolicy.dylib
-  libmscordacore.dylib
-  libmscordbi.dylib
-  libsos.dylib
-  libuv.dylib

RHEL 7.2

-  dotnet-sdklist
-  dotnet-sdklist.deps.json
-  dotnet-sdklist.dll
-  dotnet-sdklist.pdb
-  dotnet-sdklist.runtimeconfig.json

-  libclrjit.so
-  libcoreclr.so
-  libcoreclrtraceptprovider.so
-  libdbgshim.so
-  libhostfxr.so
-  libhostpolicy.so
-  libmscordacore.so
-  libmscordbi.so
-  libsos.so
- libsosplugin.so

Operating System specific assemblies



Indirection between assemblies and 3rd party libraries

Shims offer flexibility for OS variations of dependencies

System.*.Native and libcoreclr PAL

Versioning of .NET Core

Different versions of .NET Core installed side-by-side

Long Term Support (LTS): Patches include fixes and updates, no new features

Current: New features and fixes. Might require rework

Roll-forward policy

Apps automatically use higher patch version (e.g. 1.0.0 -> 1.0.1)

Major and minor versions (e.g. 1.1, 2.0) require explicit upgrade (1.1, 2.0)

Choose required version by updating metapackage

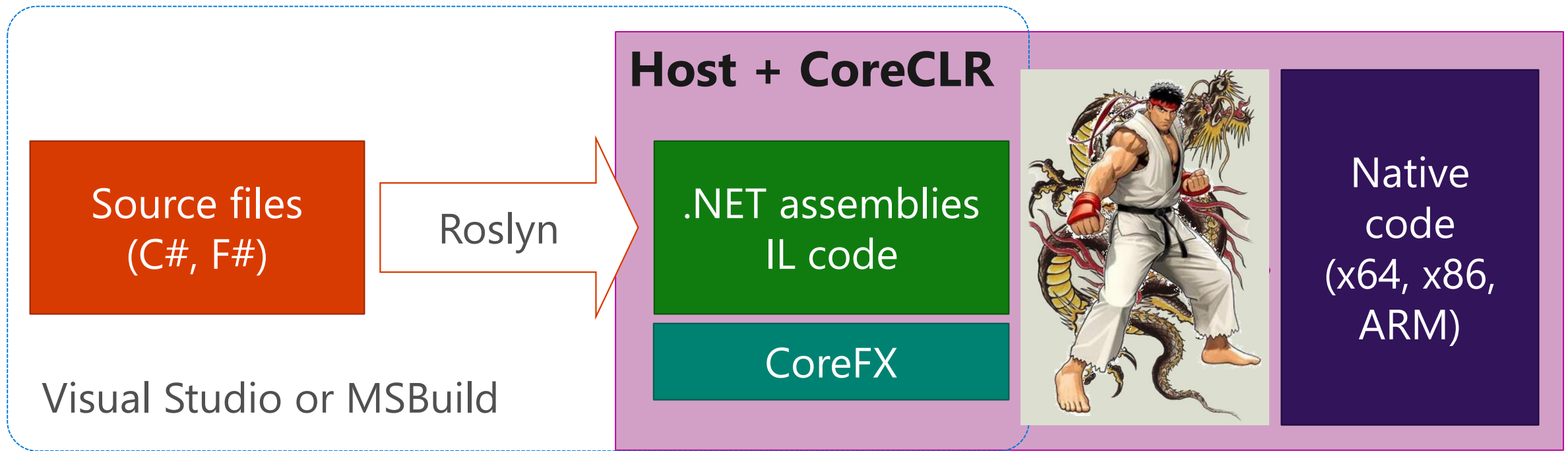
```
<PackageReference Include="Microsoft.NETCore.App">  
  <Version>1.0.1</Version>  
</PackageReference>
```

Application frameworks need newer NuGet packages

ASP.NET Core, EF Core, ...

.NET Core compilation models

Just-in-time (JIT) compilation



Compile time

Runtime

* = Experimental

.NET Core compilation models

Ahead-Of-Time (AOT) compilation

Produces single, standalone binary

Includes specialized runtime

Leverages code optimization techniques

- LLVM (for LLILC)

- Visual Studio or MSBuild, compilers and linkers

Source files
(C#, F#)

Roslyn

.NET Core app
assemblies
(IL code)

CoreFX

RyuJIT
LLILC*
IL to C++*

Native code
(x64, x86,
ARM)

CoreRT

Compile time

Runtime

* = Experimental

.NET Native AOT

Similar to cloud compiler for UWP apps submitted to Windows Store

No JIT compiler for .NET Native on UWP or Xamarin iOS tool chain

Single binary

Simplest copy deployment

Easily create Docker image

Highly optimized machine code during build

Support for native executables initially

ASP.NET Core in future

No debugging support in CoreRT

Debugging support best in CoreCLR

Demo

.NET Native compilation

.NET Standard in a nutshell

.NET Standard is a specification

Represents a set of APIs that all .NET platforms have to implement
.NET Core is an implementation of the .NET Standard

```
public partial class Object
{
    public Object() { }
    public virtual bool Equals(System.Object obj) { throw null; }
    public static bool Equals(System.Object objA, System.Object objB) { throw null; }
    ~Object() { }
    public virtual int GetHashCode() { throw null; }
    public System.Type GetType() { throw null; }
    protected System.Object MemberwiseClone() { throw null; }
    public static bool ReferenceEquals(System.Object objA, System.Object objB) { throw null; }
    public virtual string ToString() { throw null; }
}
```

.NET Standard in more detail

Represents .NET BCL

Targeted towards libraries

Replaces Portable Class Library
as tooling for multi-platform libraries

Contains APIs that are
guaranteed to work everywhere

Systematic approach to
versioning

The screenshot shows the 'General' and 'Targeting' tabs of a project's properties window. In the 'General' tab, the 'Assembly name' and 'Default namespace' are both set to 'DotNextLibrary'. The 'Output type' is set to 'Class Library'. There is a button labeled 'Assembly Information...'. In the 'Targeting' tab, the 'Targets' list includes '.NET Framework 4.5', 'ASP.NET Core 1.0', 'Windows 8', 'Windows Phone 8.1', and 'Windows Phone Silverlight 8'. There is a 'Change...' button and a link to 'Target .NET Platform Standard'.

General

Assembly name: DotNextLibrary

Default namespace: DotNextLibrary

Output type: Class Library

Assembly Information...

Targeting

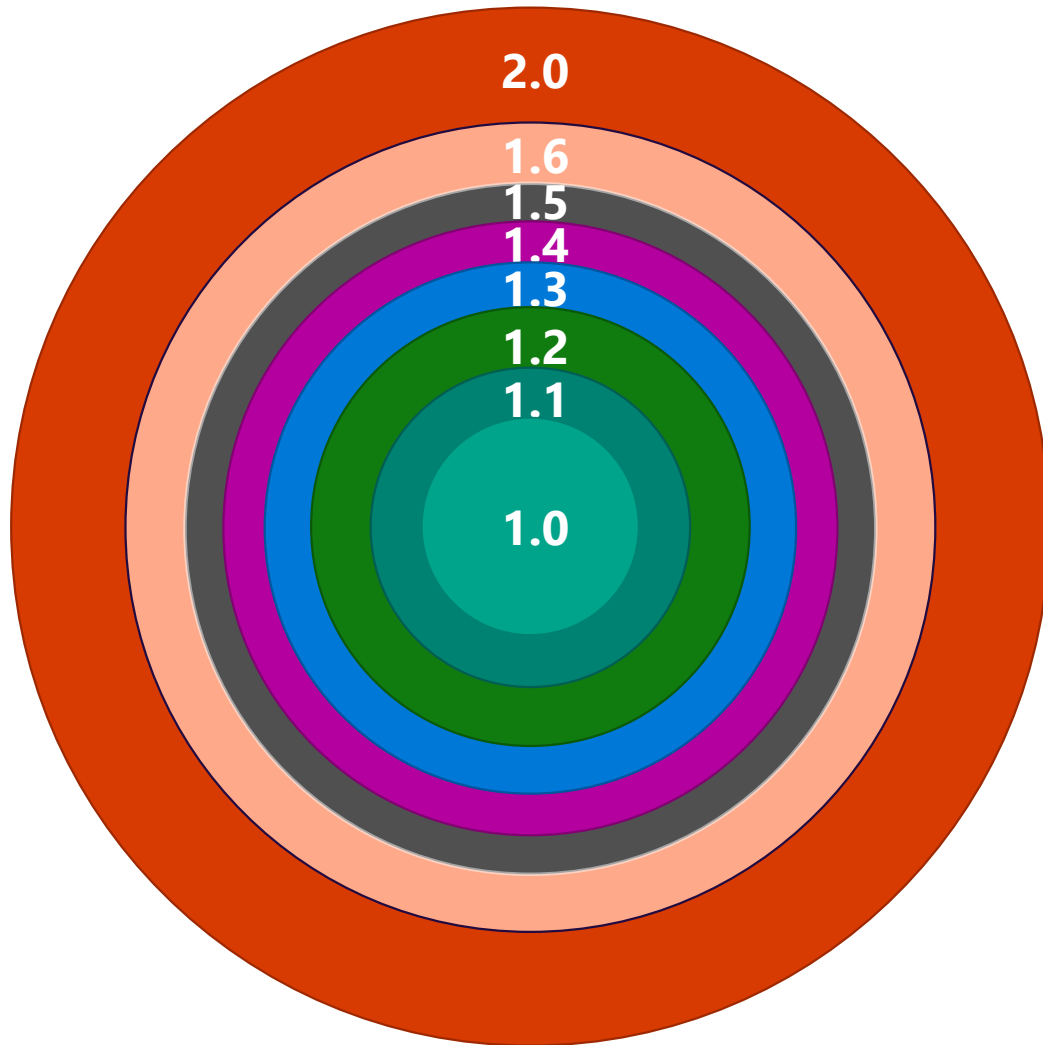
Targets:

- .NET Framework 4.5
- ASP.NET Core 1.0
- Windows 8
- Windows Phone 8.1
- Windows Phone Silverlight 8

Change...

[Target .NET Platform Standard](#)

Versioning in .NET Standard



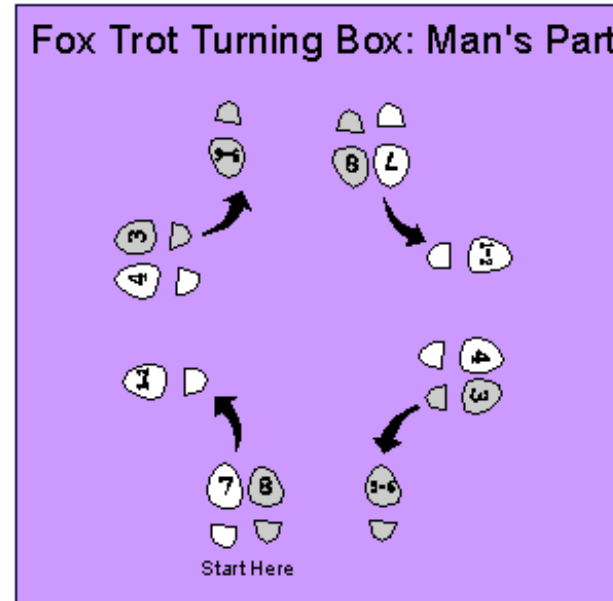
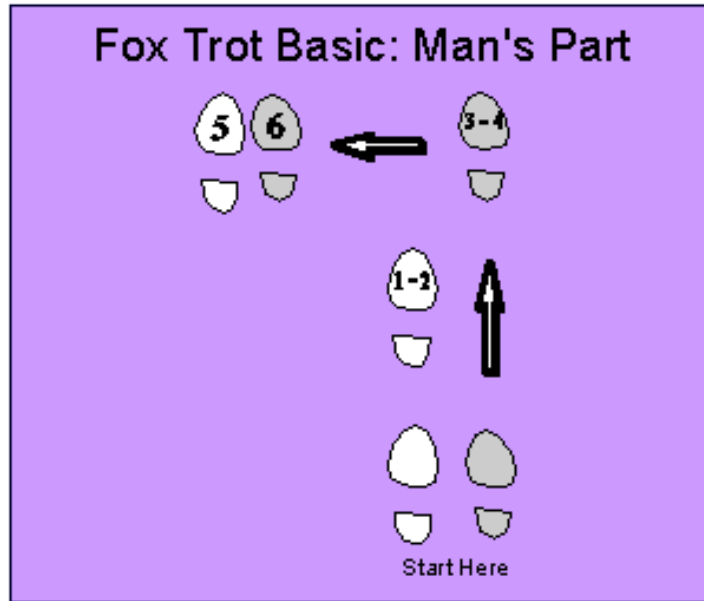
Higher versions incorporate all APIs from previous versions

Concrete .NET platforms implement a specific version of .NET Standard

.NET Platforms and Standards

.NET Platform	1.0	1.1	1.2	1.3	1.4	1.5	1.6	2.0
.NET Core	→	→	→	→	→	→	1.0	vNext
.NET Framework	→	4.5	4.5.1	4.6	→	→	→	4.6.1
Mono	→	→	→	→	→	→	4.6	vNext
Xamarin.iOS	→	→	→	→	→	→	10.0	vNext
Xamarin.Android	→	→	→	→	→	→	7.0	vNext
Universal Windows Platform	→	→	→	→	10.0	→	→	vNext
Windows	→	8.0	8.1					
Windows Phone	→	→	8.1					
Windows Phone Silverlight	8.0							

Foxtrot Standard 1.0, 1.1 and 2.0



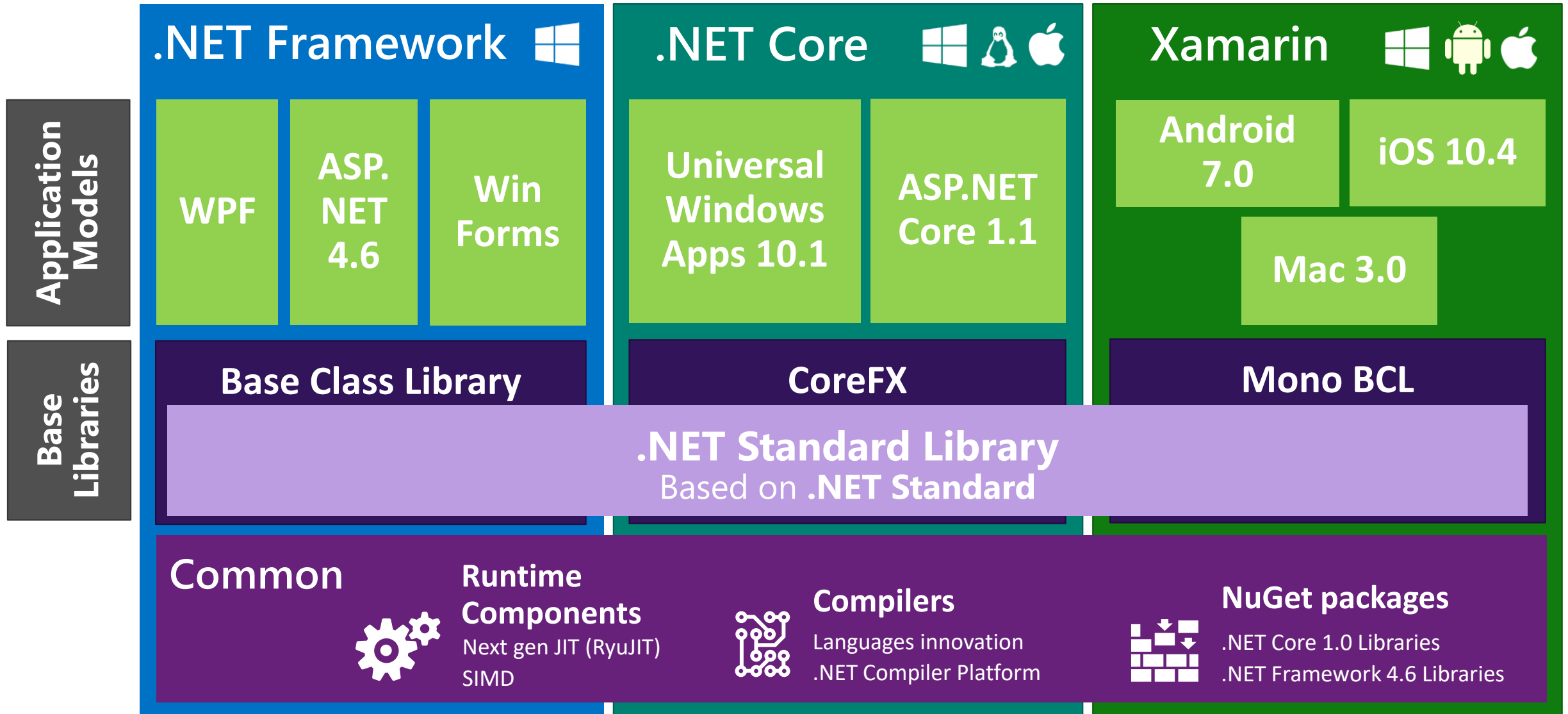
Corner
Promenade with
underarm turn
Corte
Twinkle

Foxtrot 1.0:
Basic steps

Foxtrot 1.1:
Basic steps +
Box turn

Foxtrot 2.0:
Foxtrot 1.1 +
Many steps +
Compat with Waltz

Revisiting .NET Platforms



.NET Standard 2.0

Introduces more APIs

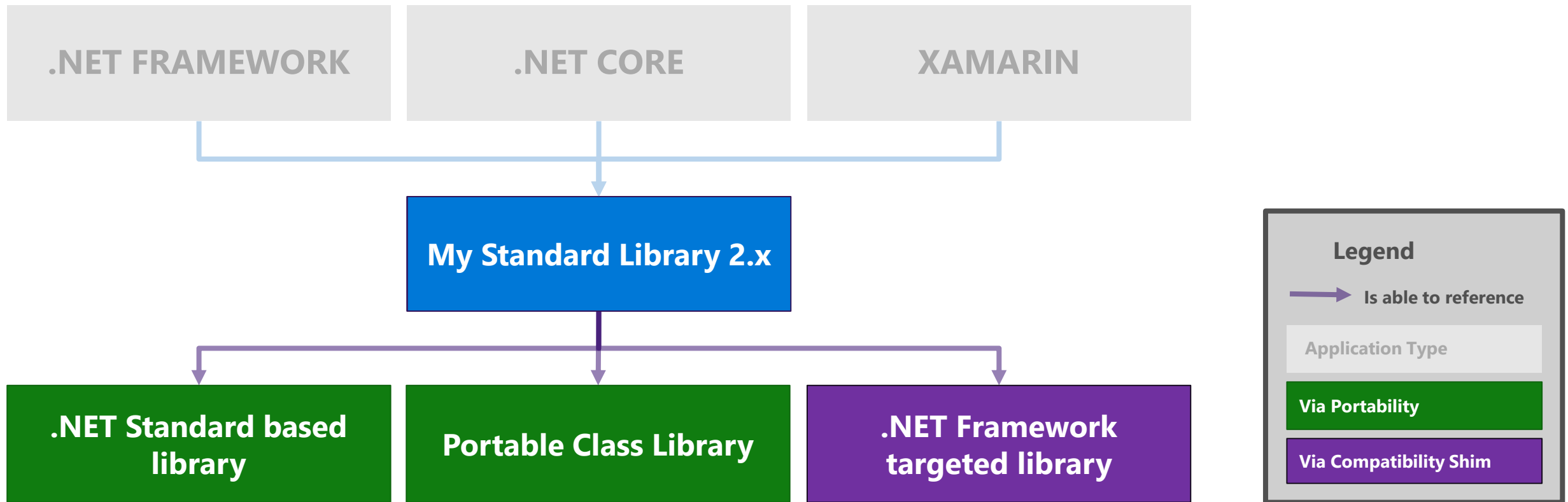
Version	#APIs	Growth %
1.x	13,501	+1%
2.0	32,638	+142%

Creates compatibility
with .NET Framework
libraries

Compatibility shim bridges .NET Standard
based libraries to .NET FX libraries (mscorlib)

XML	XLinq • XML Document • XPath • Schema • XSL
SERIALIZATION	BinaryFormatter • Data Contract • XML
DATA	Abstractions • Provider Model • DataSet
NETWORKING	Sockets • Http • Mail • WebSockets
IO	Files • Compression • MMF
THREADING	Threads • Thread Pool • Tasks
CORE	Primitives • Collections • Reflection • Interop • Linq

Referencing libraries from .NET Standard



Porting existing code to .NET Core

1. Identify projects to port
2. Understand external dependencies
3. Change to .NET 4.6.1
4. Recompile
5. Run ApiPort tool
6. Change code to solve issues

Discontinued areas:

App Domains

.NET Remoting

Sandboxing (Code Access Security)

Replacements where sensible

Tools

<https://packagesearch.azurewebsites.net>

<https://github.com/Microsoft/dotnet-apiport>

Demo Testing .NET Standard compatibility with ApiPort



The screenshot shows a Counter-Strike: Source HLTV demo interface. At the top left is the 'The Orange Box' logo. At the top right, it displays 'Counter-Terrorists : 0', 'Terrorists : 1', 'Spectators 6525', and a timer '0:41'. The background is a first-person view of a player holding a handgun in a stone-walled corridor. A scoreboard overlay is centered on the screen, showing the following data:

	Score	Deaths	Latency
Terrorists - 5 players	1	2	
magic.Archi	4	0	2
magic.hstw	8	2	1
magic.DEntw	1	0	8
magic.pAm <v4v>	0	1	1
magic.f0rest	0	1	2
Counter-Terrorists - 5 players	0	1	
SK K.Nordlund <8 Tezz	2	1	2
SK D.Wallenberg <walle>	1	1	2
SK Jimmy.A <allelelen>	1	1	4
SK A.Mohamed <SpawN>	0	1	8
SK R.Dahlstrom <RobbaN>	0	1	2
Spectators - 0 players			
DreamHack - HLTV Master			
WGT - Le1a			
WGT-Spizzy			
Player			
pongpan			

At the bottom left, it says 'GotFrag HLTV' and 'Powered by GameRail'. At the bottom right, it shows 'SK | K.Nordlund <8 Tezz [100]'.

Summary

.NET Core

- Allows your applications to span multiple operating systems
- Lightweight package-based and cloud/container optimized

.NET Standard

- Incremental specification of .NET API surface
- Will make your class library portable to multiple .NET platforms

Future

- .NET Standard 2.0 provides a richer API surface and better migration story
- Tooling and IDEs will catch up

Resources

<http://dot.net>

<http://github.com/dotnet/...>

corefx

lilc

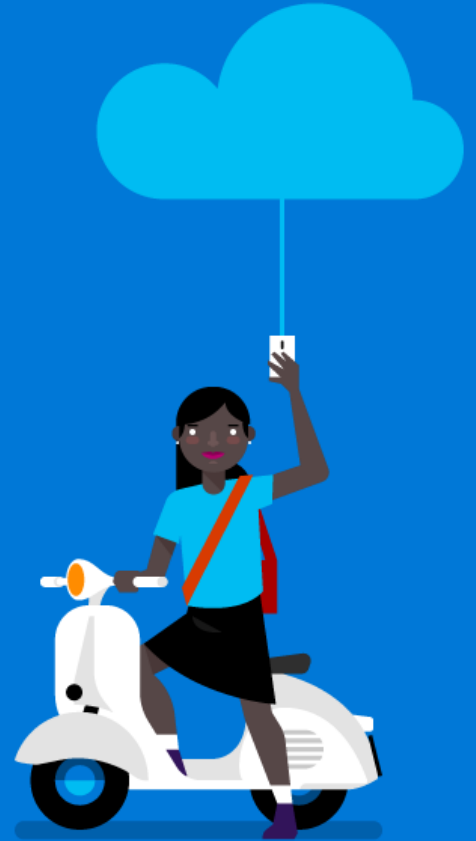
corert

standard

CLI

<http://visualstudio.com>

<https://docs.microsoft.com/en-us/dotnet/articles/core/>



Questions and Answers

Maybe later?

@alexthissen

athissen@xpirit.com

