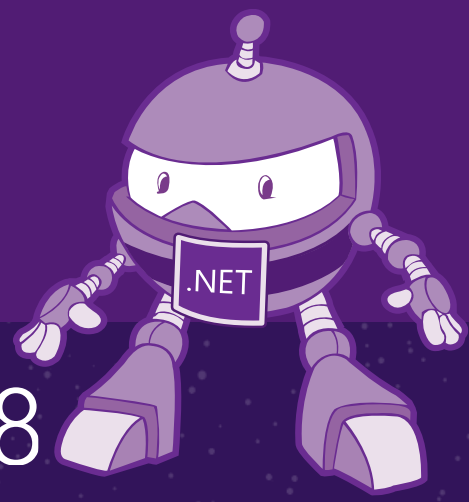

DotNet Toscana
10 Years
.NET Conf 2018
Discover the world of .NET
.NET
dotnet new
www.dotnetconf.net

Core is the new Black
Modern Framework for Modern Apps

Giancarlo Lelli
Avanade Italy / @itsonlyGianca / gcarlo.lelli@live.com



.NET Conf 2018

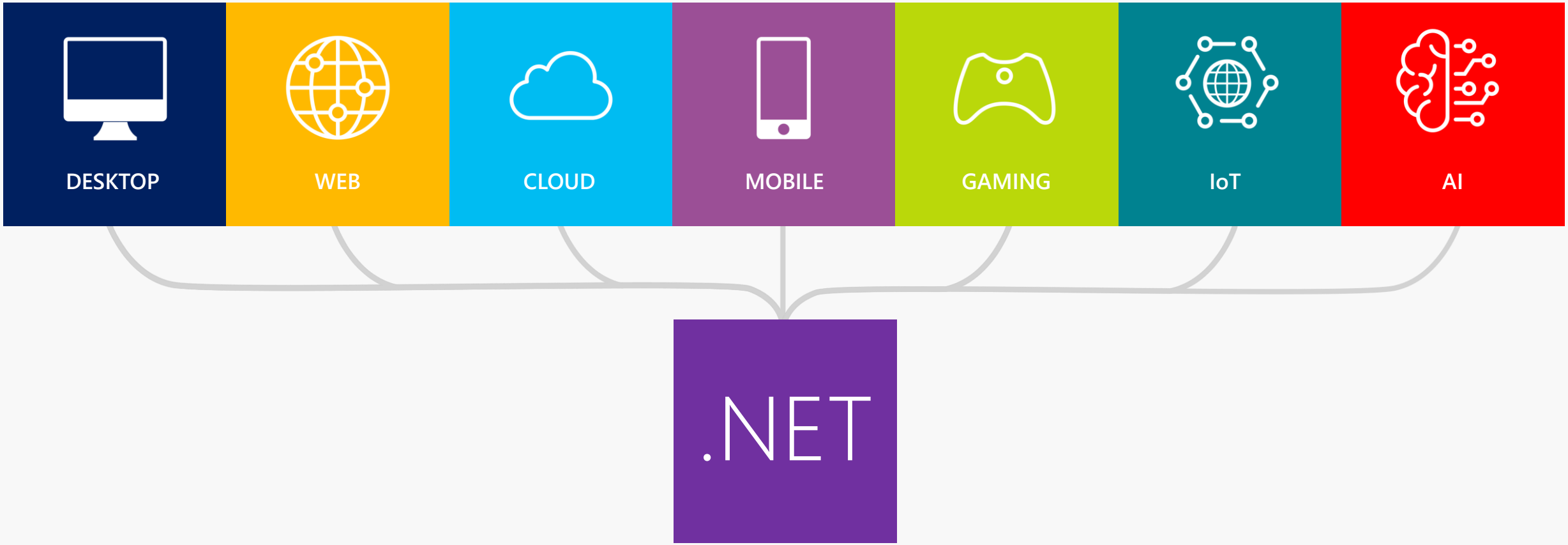
DotNet Toscana

Grazie ai nostri sponsor

DotNet Toscana

.NET Conf 2018

Your platform for building **anything**



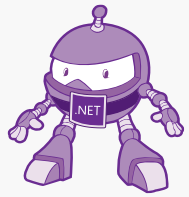
.NET Growth Continues

Visual Studio

+1 million new monthly active .NET developers in last year

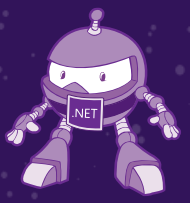
.NET Core

Over half a million .NET Core 2.0 developers

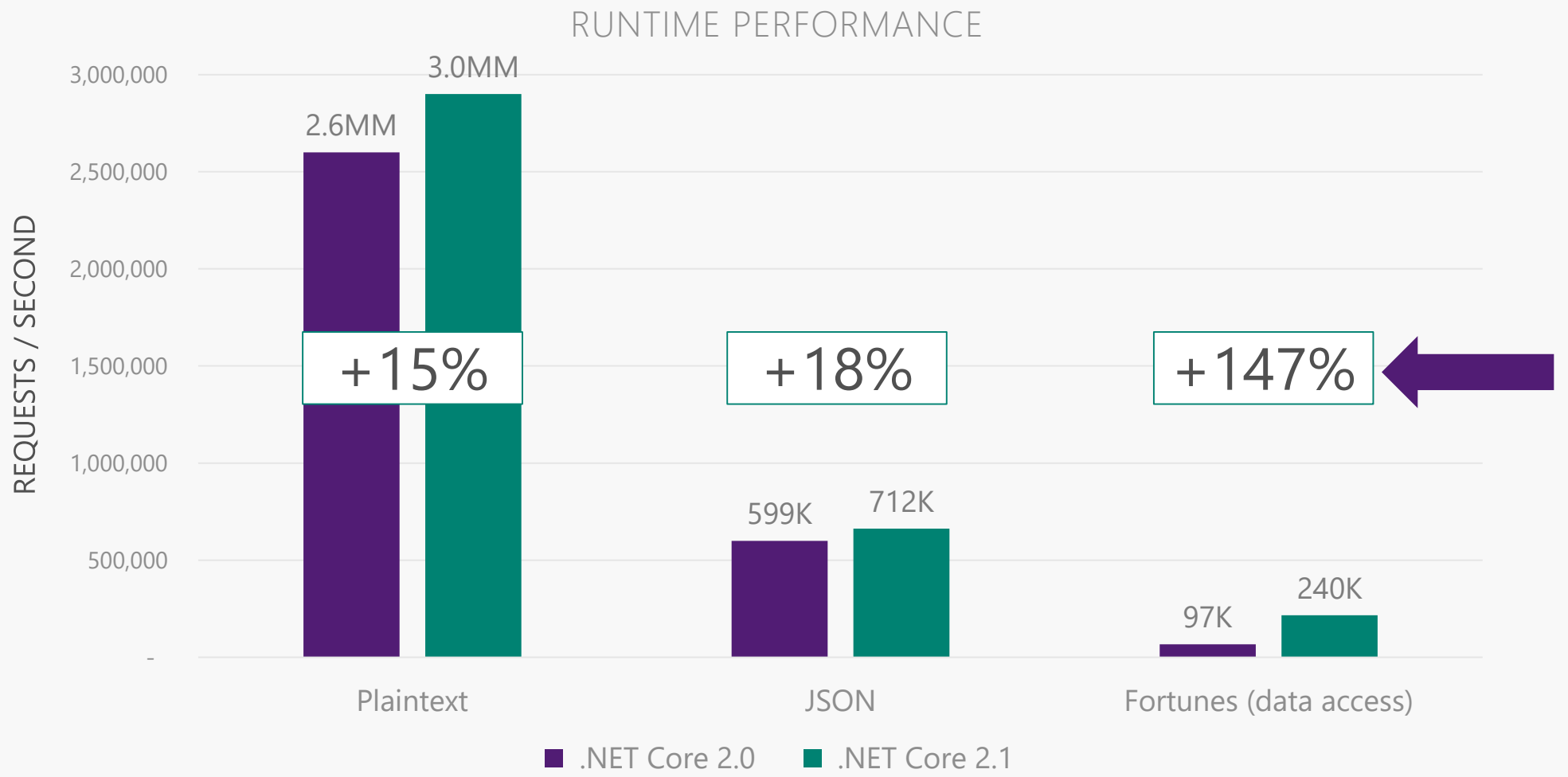


Now Available
.NET Core 2.1!

www.dot.net



.NET Core 2.1 on TechEmpower



DotNet Toscana

[Data sourced from tests on TechEmpower round 16]

.NET Conf 2018

.NET Core 2.1 Major Features

.NET Core

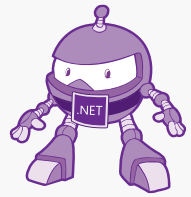
- Global Tools
- Span<T>
- Sockets
- HttpClient Performance
- Windows Compatibility Pack

EF Core

- Lazy Loading
- Value conversions
- Query types
- Data seeding

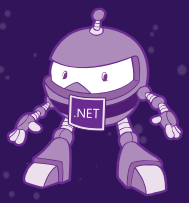
ASP.NET Core

- HTTPS
- Razor UI as a library
- HttpClientFactory
- ASP.NET Core SignalR

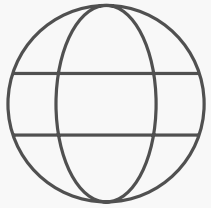


Announcing .NET Core 2.2 Preview 2

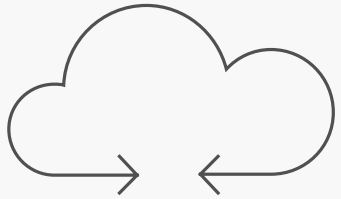
https://aka.ms/DotNetCore22



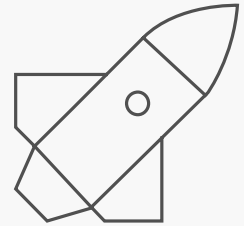
.NET Core 2.2 Themes



Improved Web API Development



Microservices and Azure



Continued Performance Improvements

.NET Core 2.2 Major Features

.NET Core

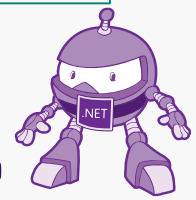
- Multi-tier JIT compilation
- SQL Connection token auth

EF Core

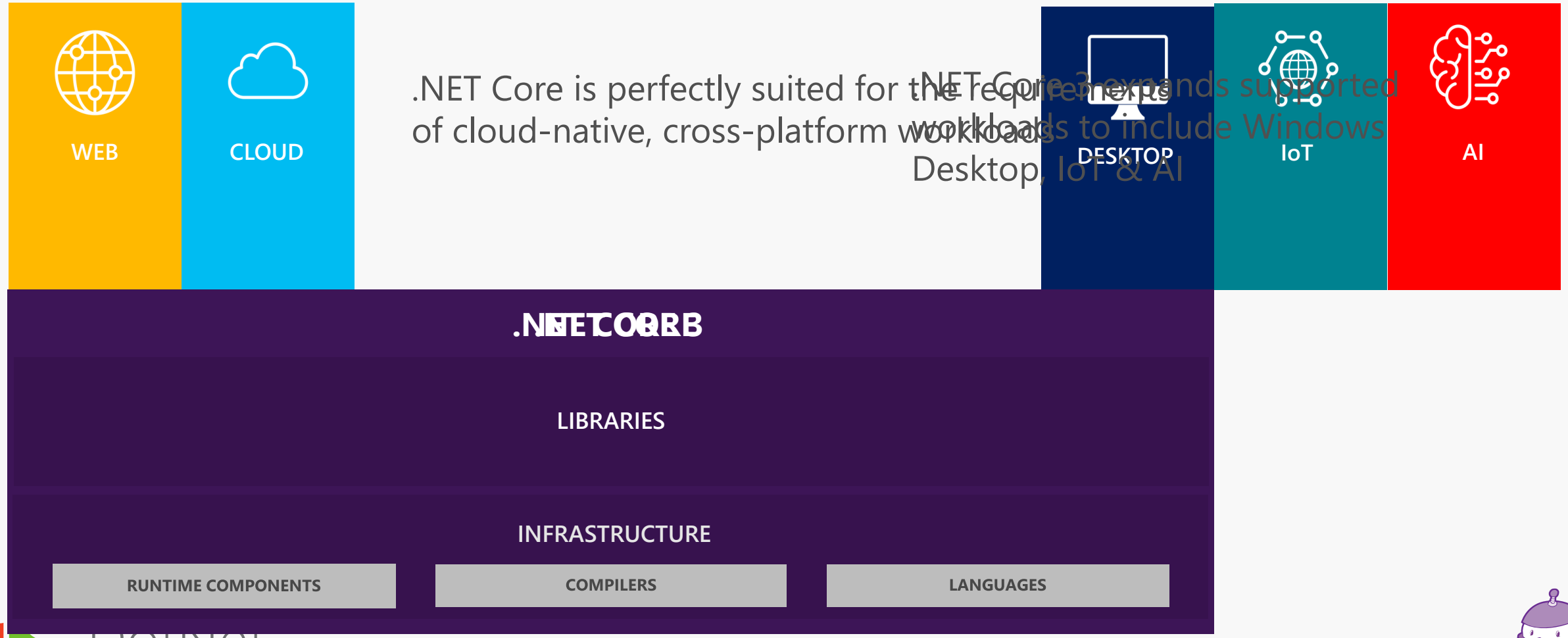
- Cosmos DB provider
- Spatial extensions for SQL Server and SQLite providers
- Reverse engineering of database views
- Collections of owned entities
- Query tagging

ASP.NET Core

- Template updates: Bootstrap 4, Angular 6
- Web API improvements, including API security
- HTTP/2
- IIS in-process hosting
- Health checks
- Endpoint routing
- SignalR Java client

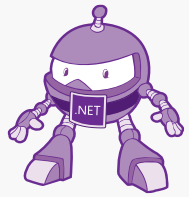


.NET Core 3


WEB
CLOUD
.NET Core is perfectly suited for the requirements of cloud-native, cross-platform workloads
.NET Core 3 expands supported workloads to include Windows Desktop, IoT & AI
DESKTOP
IoT
AI
.NET Core CLR
LIBRARIES
INFRASTRUCTURE
RUNTIME COMPONENTS
COMPILERS
LANGUAGES

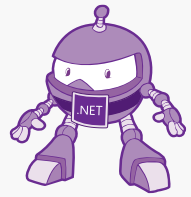
.NET Core 3 Desktop Improvements

- .NET Core 3 supports WinForms and WPF frameworks
  - XAML Islands - WinForms & WPF can host UWP
  - XAML Controls – WinForms & WPF browser and media UWP controls
  - High DPI fixes for WinForms
- Access to all the Windows 10 API's
- NET Core App Bundler
  - Precompiled, fast startup
  - Small apps by removing unused dependencies, link away unused IL
  - Single self-contained .exe



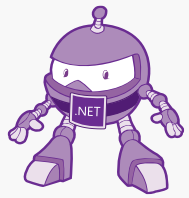
Why Windows Desktop on .NET Core?

- Side by side
- Machine global or app local framework
- Core runtime and API improvements
- SDK based .csproj project system



Update on .NET Core 3 since Build 2018

- WinForms support available in nightly builds
  - https://github.com/dotnet/core-sdk
  - dotnet new winforms
- Visual Studio supports building and debugging
  - Designers not available yet
- Publish Preview later this year

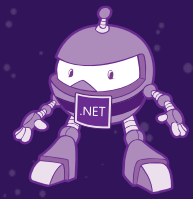


ASP.NET Core 2.2 Preview 2

.NET

DotNet Toscana

.NET Conf 2018



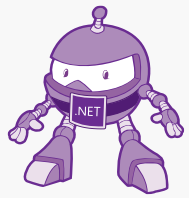
Get started with ASP.NET Core 2.2 Preview 2

Install the .NET Core 2.2 Preview 2 SDK

https://www.microsoft.com/net/download/dotnet-core/2.2

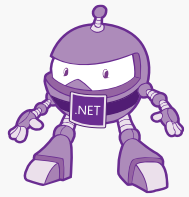
Install Visual Studio 2017 15.9 Preview 2

https://visualstudio.com/preview/

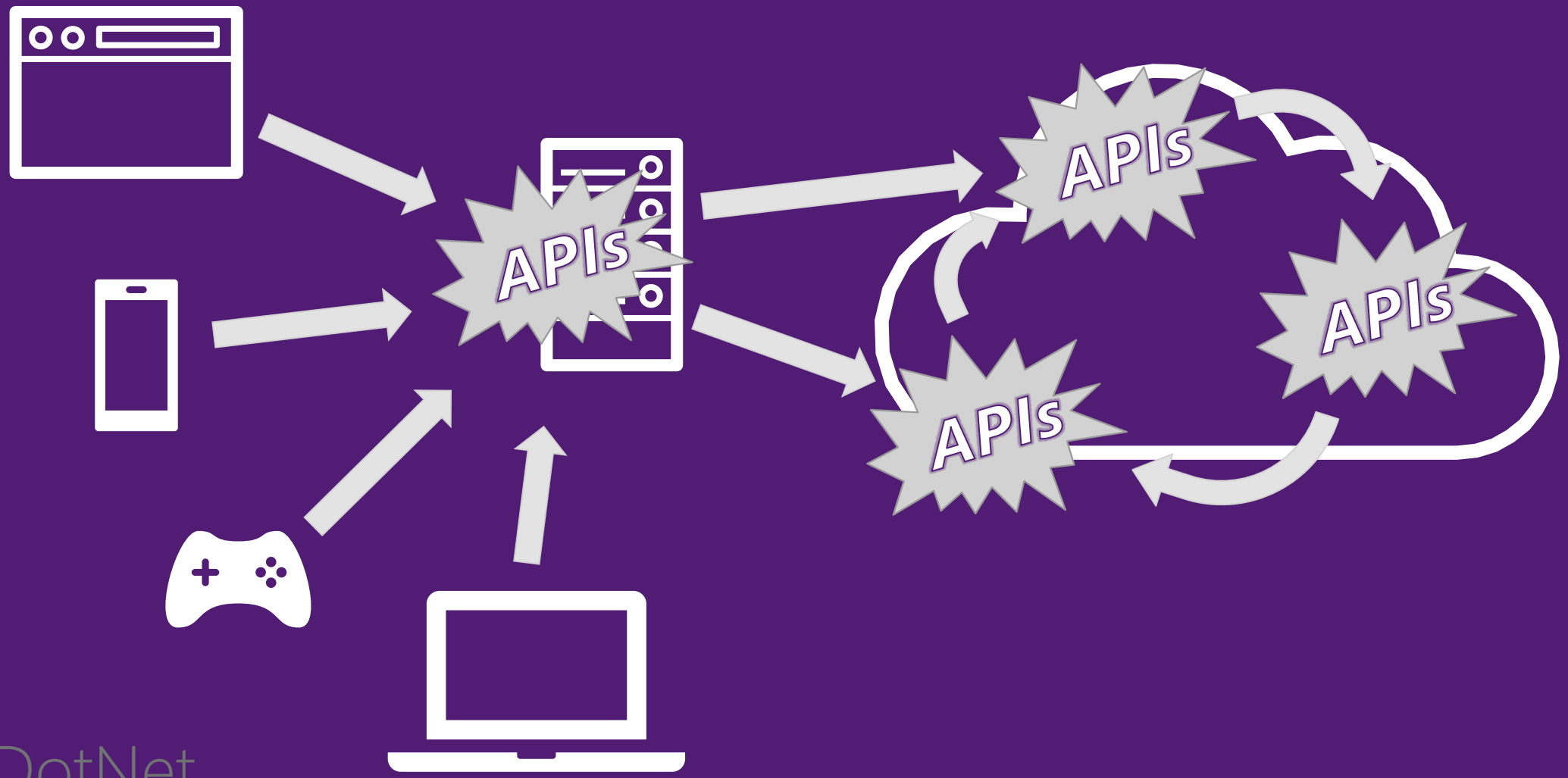


ASP.NET Core 2.2 features

- Template updates: Bootstrap 4, Angular 6
- Web API improvements
- HTTP/2
- IIS in-process hosting
- Health checks
- Endpoint routing
- SignalR Java client



APIs are everywhere!

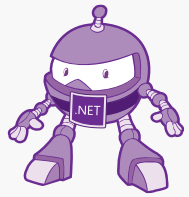


DotNet Toscana

.NET Conf 2018

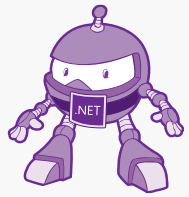
Web API improvements in ASP.NET Core 2.2

Easier to **create**	→ API Scaffolding
Easier to **test & debug**	→ HTTP REPL, Problem Details
Easier to **document**	→ API conventions & analyzer
Easier to **consume**	→ Code generation (Preview 3)
Easier to **secure**	→ Web API security (Preview 3)
Easier to **monitor**	→ Health checks integration
Improved **performance**	→ HTTP/2, endpoint routing, IIS in-proc hosting



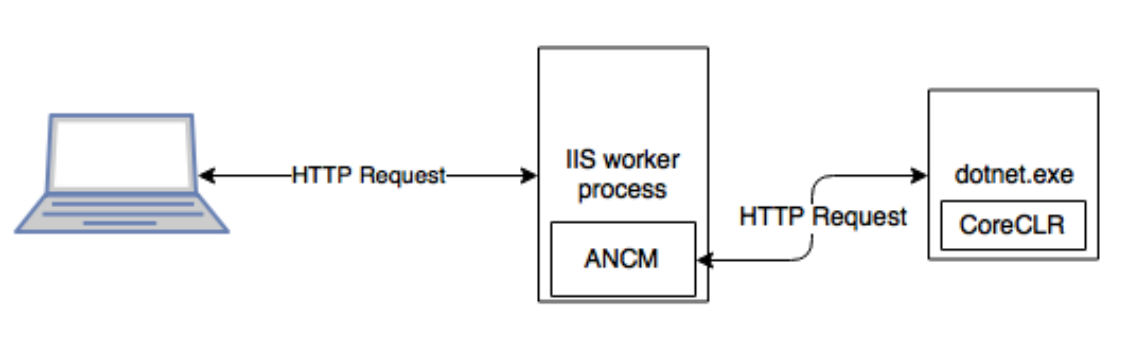
HTTP/2

- Now available cross-platform in Kestrel
- Application-Layer Protocol Negotiation (ALPN)
- Header compression
- Multiplexed streams over same connection
- Some limitations
  - Server push, stream prioritization not currently supported
  - Not supported for edge use at this time

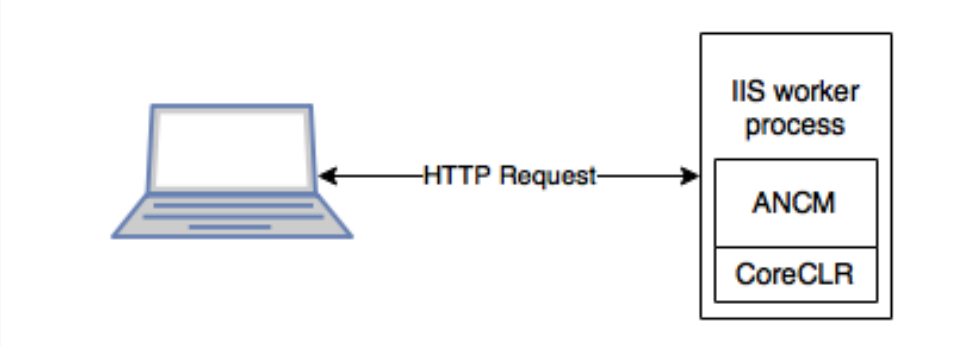


IIS in-process hosting

Out of process (current)

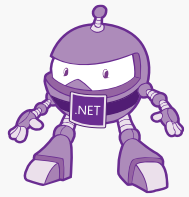


In process (NEW!)

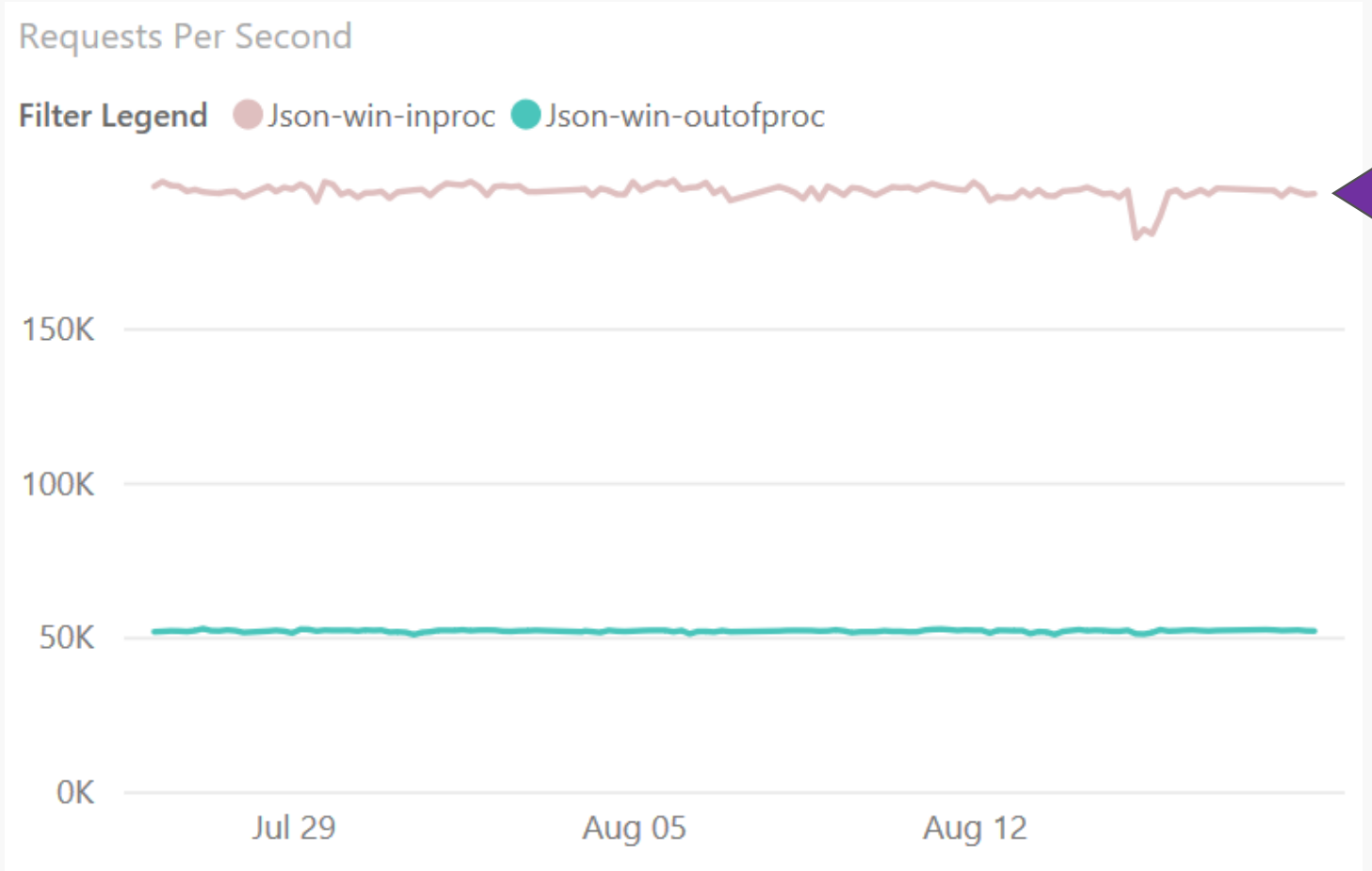


IIS in-process hosting

- Improved performance, reliability, and diagnostics



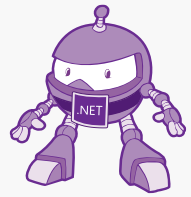
IIS in-process hosting – Performance!



4x faster!

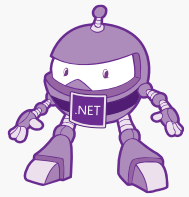
Health checks

- Add dedicated health endpoints to your application
- Integrate with container orchestrators and load balancers
- Support liveness and readiness probes



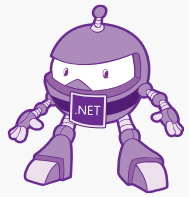
Endpoint routing

- New routing implementation
  - On by default for 2.2 with compatibility switches
- Better throughput (~10%) and scalability
- Link to endpoints from outside of MVC
- Various minor improvements
  - Parameter transformers, new catch all syntax {**path}



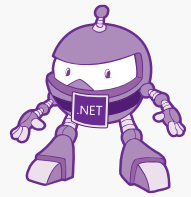
ASP.NET Core 2.2 features

- Template updates: Bootstrap 4, Angular 6
- Web API improvements
- HTTP/2
- IIS in-process hosting
- Health checks
- Endpoint routing
- SignalR Java client



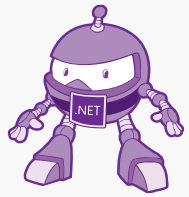
ASP.NET Core 2.2 schedule

- Preview 2 - Sept
- Preview 3 – Oct
- RTW – Year-end 2018



SignalR Java client

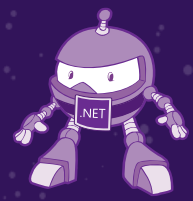
- Connect to ASP.NET Core SignalR hubs in Java
- Available via Gradle and Maven
- Now supports Azure SignalR Service
- Complete Android sample
  - https://github.com/aspnet/SignalR-samples/tree/master/AndroidJavaClient



What is .NET Standard?

DotNet Toscana

.NET Conf 2018



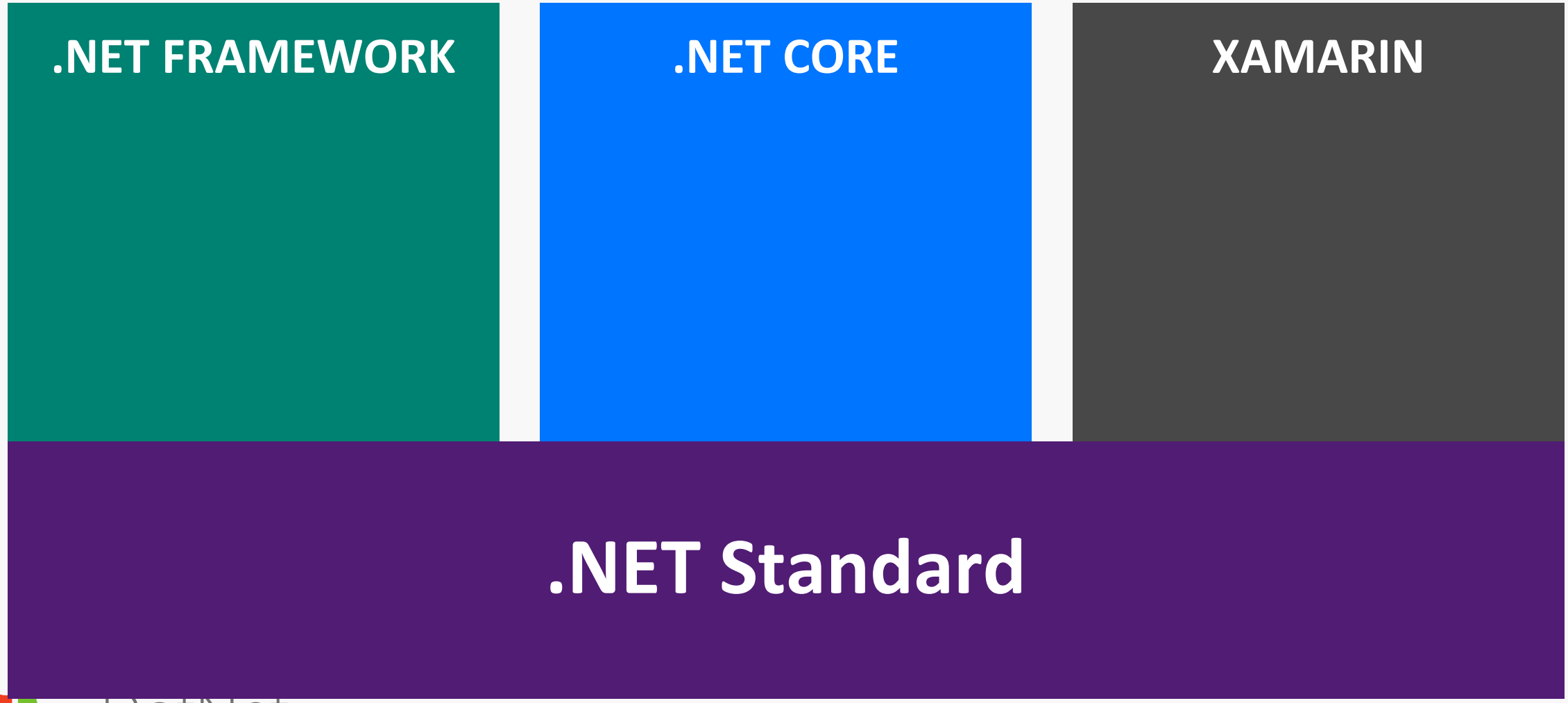
What is .NET Standard?

- .NET Standard is a specification
- A set of APIs all .NET platforms have to implement

.NET Standard	~	HTML specification
.NET Core .NET Framework Xamarin	~	Browsers

```
$ dotnet new classlib -o My.Class.Library
```

.NET Standard in context



What's in .NET Standard?

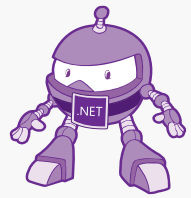
Microsoft.Win32.SafeHandles
System
System.CodeDom.Compiler
System.Collections
System.Collections.Concurrent
System.Collections.Generic
System.Collections.ObjectModel
System.Collections.Specialized
System.ComponentModel
System.ComponentModel.D
System.
System.
System.
System.
System.D
System.D
System.D
System.D
System.D
System.Diagnostics.Tracing
System.Drawing
System.Dynamic
System.Globalization
System.IO
System.IO.Compression
System.IO.IsolatedStorage
System.IO.MemoryMappedFiles
System.IO.Pipes
System.Linq
System.Linq.Expressions
System.Net
System.Net.Cache
System.Net.Http
System.Net.Http.Headers
System.Net.M
System.R
System.Runtime
System.Runtime.CompilerServices
System.Runtime.ConstrainedExecution
System.Runtime.ExceptionServices
System.Runtime.InteropServices
System.Runtime.InteropServices.ComTypes
System.Runtime.Serialization
System.Runtime.Serialization.Formatters
System.Runtime.Serialization.Formatters.Binary
System.Runtime.Serialization.Json
System.Runtime.Versioning
System.Security
System.Security.Authentication
System.Security.Authentication.ExtendedProtection
System.Security.Claims
tificates
System.Transactions
System.Windows.Input
System.Xml
System.Xml.Linq
System.Xml.Resolvers
System.Xml.Schema
System.Xml.Serialization
System.Xml.XPath
System.Xml.Xsl

All the foundational APIs
~37k APIs in .NET Standard 2.0

Versions of .NET Standard

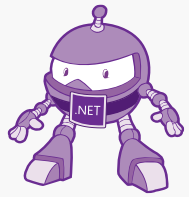
.NET Standard	1.0	1.1	1.2	1.3	1.4	1.5	1.6	2.0
.NET Core	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0
.NET Framework [1]	4.5	4.5	4.5.1	4.6	4.6.1	4.6.1	4.6.1	4.6.1
Mono	4.6	4.6	4.6	4.6				
Xamarin.iOS								
							10.0.16299	10.0.16299
		8.0	8.1					
Windows Phone	8.1	8.1	8.1					
Windows Phone Silverlight	8.0							

Don't worry about it.
Start with .NET Standard 2.0.



.NET Standard is also Open Source!

- Anybody can propose API additions
- The review board approves the API
  - Has representatives from the .NET Foundation, Microsoft, Xamarin/Mono, & Unity
- Acceptance requires
  - A stable implementation that is shipped in at least one .NET implementation
  - Sponsorship from a board member
- Next version is planned here:
  - https://github.com/dotnet/standard/tree/master/docs/planning/netstandard-2.1

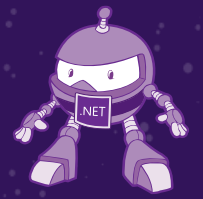


Using platform-specific APIs from .NET Standard

.NET

DotNet Toscana

.NET Conf 2018

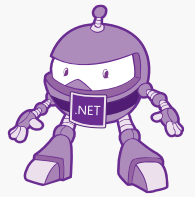


Windows Compatibility Pack

- Microsoft.Windows.Compatibility (NuGet package)
  - Can be referenced from .NET Core as well as from .NET Standard
  - Has ~21k APIs (Windows-only as well as cross-platform)

- Contents

ACLs	Drawing	Ports
Code Pages	EventLog	Registry
CodeDom	MEF v1	Runtime Caching
Configuration	Odbc	WCF
Crypto	Perf Counters	Windows Services
DirectoryServices	Permissions	…



Detecting usage of unsupported APIs

- Use API Analyzer!
  - https://aka.ms/apianalyzer
  - Roslyn analyzer that flags usages of APIs that don't work across all platforms

```
private static string GetLoggingPath()
{
    using (var key = Registry.CurrentUser.OpenSubKey(@"Software\Fabrikam\AssetManagement"))
    {
        if (key?.GetValue("LoggingDirectoryP
            return configuredPath;
    }

    var appDataPath = Environment.GetFolderP                    pplicationData);
    return Path.Combine(appDataPath, "Fabrik
}
```

RegistryKey RegistryKey.OpenSubKey(string name) (+ 4 overloads)
Retrieves a subkey as read-only.

Exceptions:
ArgumentNullException
ObjectDisposedException
System.Security.SecurityException

RegistryKey.OpenSubKey(string) isn't supported on Linux, MacOSX

Show potential fixes (Alt+Enter or Ctrl+.)

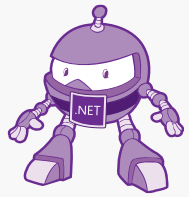
Error List

Entire Solution | 0 Errors | 2 Warnings | 0 Messages | Build + IntelliSense | Search Error List

	Code	Description	Project	File	Line	Suppression St...
⚠	PC001	RegistryKey.GetValue(string) isn't supported on Linux, MacOSX	Fabrikam.Shared	Logger.cs	19	Active
⚠	PC001	RegistryKey.OpenSubKey(string) isn't supported on Linux, MacOSX	Fabrikam.Shared	Logger.cs	17	Active

Multi-Targeting Best Practices

- DO start with .NET Standard 2.0
  - Most general purpose libraries will not need APIs outside this set.
- CONSIDER targeting multiple frameworks
  - If you need to call platform-specific APIs outside of .NET Standard
- DO NOT drop support for .NET Standard
  - Instead, throw from the implementation and offer capability APIs. This way, your library can used anywhere and supports runtime light-up.
- DO share your component using a NuGet package
  - It shields consumers from having to pick the appropriate implementation.
- CONSIDER using MSBuild.Sdk.Extras
  - Thanks Oren! ☺



.NET Standard &
.NET Framework

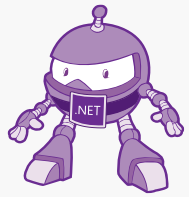
DotNet Toscana

.NET Conf 2018

Sorry.

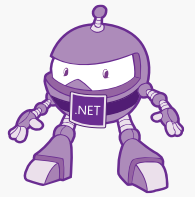
- We retroactively made .NET Framework 4.6.1 support .NET Standard 2.0
  - Turns out this was a mistake as it has a tail of issues.
  - Lesson learned for us: after a .NET platform has shipped, the version of .NET Standard it supports should be considered immutable.

- .NET Framework 4.7.2 is the recommended version to consume .NET Standard 1.5+
  - Application authors: consider upgrading
  - Library authors: consider multi-targeting for .NET Framework 4.6.1



Strong naming

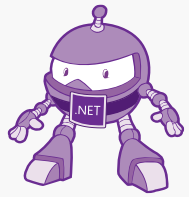
- Key signing to give assemblies an additional piece for the name
  - MyLibrary, Version=1.0.0.0, Culture=neutral, PublicKeyToken=70d3c3735d2363d4
  - Don’t confuse strong names with Authenticode signing, the industry standard based on certificates for indicating who authored a specific binary
- Disambiguates assemblies with the same simple name (such as “Calculator”)
  - Required for GAC or for loading assemblies side-by-side



Strong naming

- Strong naming is generally viral
  - Strong named assemblies can only reference strong named assemblies
- Adding or removing a strong name is a binary breaking change

- We recommend that you:
  - Strong name your libraries from the get-go
  - Check in the public and private key into your OSS repos
  - Do not make security decisions based on the strong name



Binding redirects

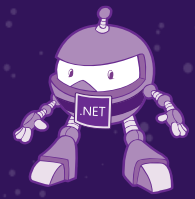
- A .NET Framework-only concept
  - Only loads assemblies where the referenced version matches the version on disk
  - Problematic when someone references an older version while your app deploys a higher version
- To accept a higher version, your app.config needs to explicitly allow for that by adding a redirect:

```
<assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1" appliesTo="v1.0.3705">
  <dependentAssembly>
    <assemblyIdentity name="System.Collections.Immutable"
                      publicKeyToken="b03f5f7f11d50a3a“
                      culture="neutral"/>
    <bindingRedirect oldVersion="0.0.0.0-1.2.2.0" newVersion="1.2.2.0"/>
  </dependentAssembly>
</assemblyBinding>
```

Tips

DotNet Toscana

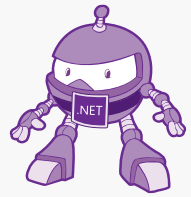
.NET Conf 2018



Versioning

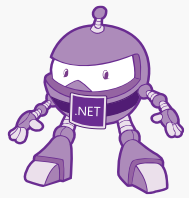
You need to know about four version numbers:

Kind	When to increment	Comment
Package Version	Every change	The ID of the NuGet package.
Assembly Version	As you see fit	The version number of the assembly. Used by the loader to resolve assemblies.
File Version	Every change	Generic concept, used by installers to determine which file is newer.
Informational Version	As you see fit	Display string, doesn't need to be a version.



Best Practices

- DO follow the API Design Guidelines
- DO target .NET Standard 2.0
- CONSIDER using multi-targeting to allow for platform-specific code
  - **CONSIDER** dual-targeting for .NET Framework 4.6.1
  - **DO** use NuGet for packaging multi-targeted libraries
  - **DO** throw PlatformNotSupportedException for unsupported APIs
  - **CONSIDER** offering capability APIs so that consumers can check upfront
- DO strong name your libraries
  - **AVOID** if your library can't be used on .NET Framework
  - **DO** check in the public & private key

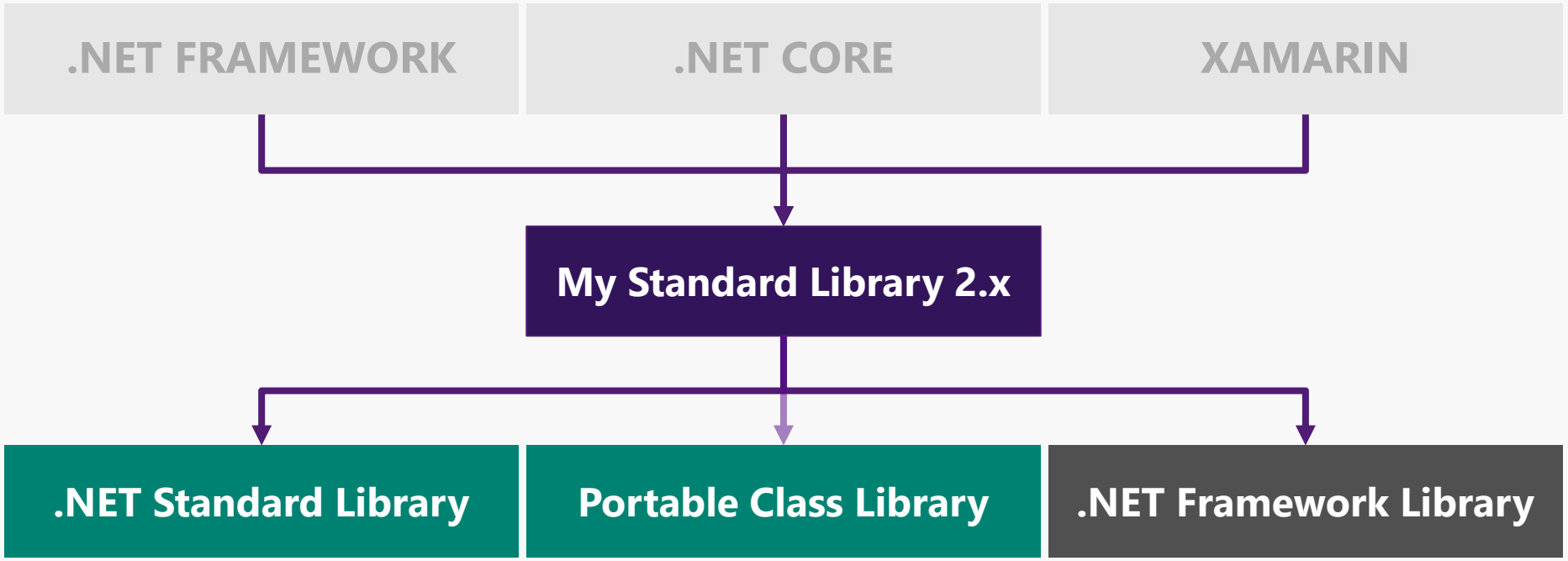


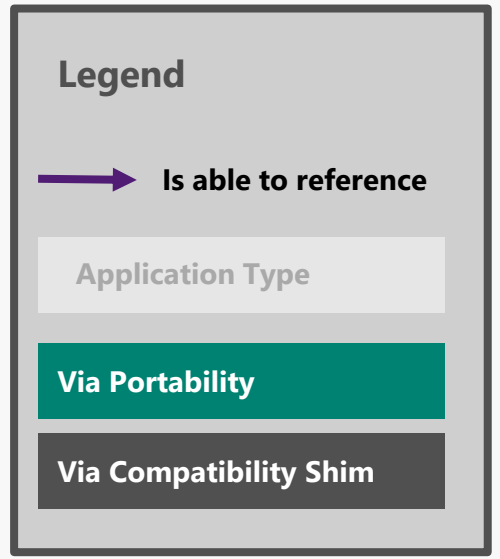
Under the hood

DotNet Toscana

.NET Conf 2018

What can you reference from .NET Standard?

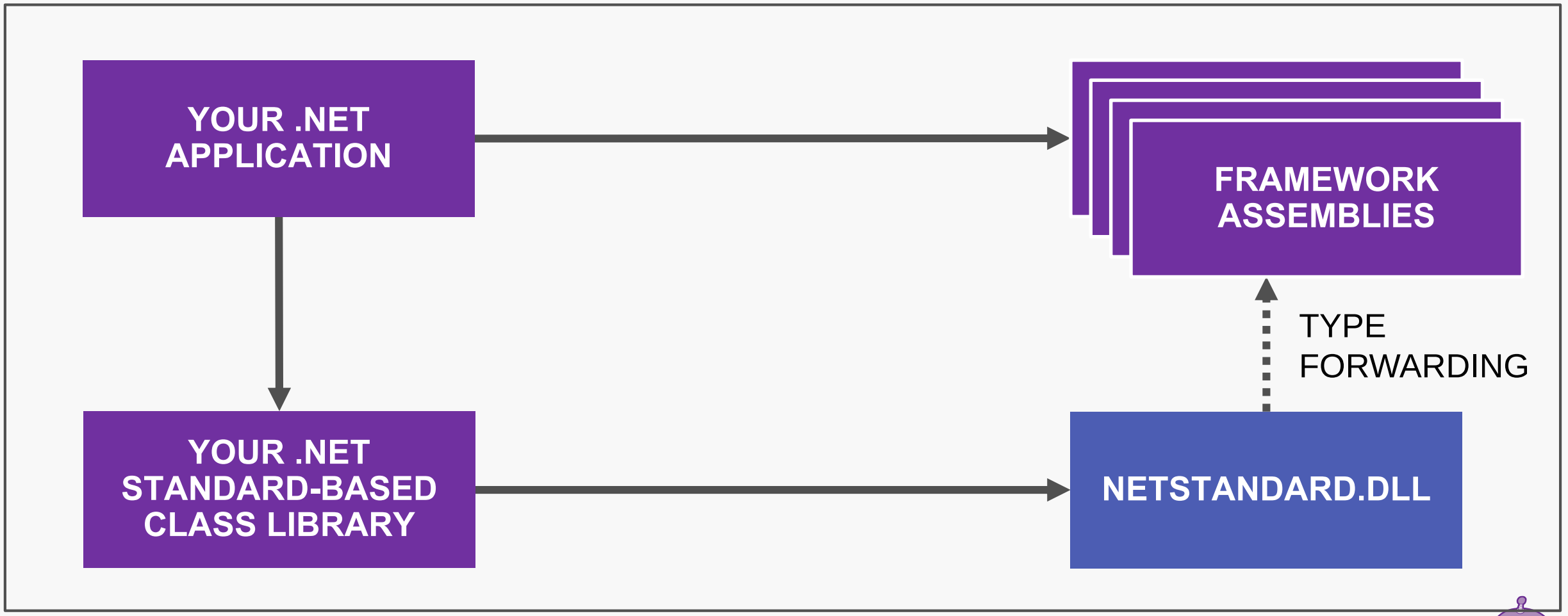




.NET Standard under the hood

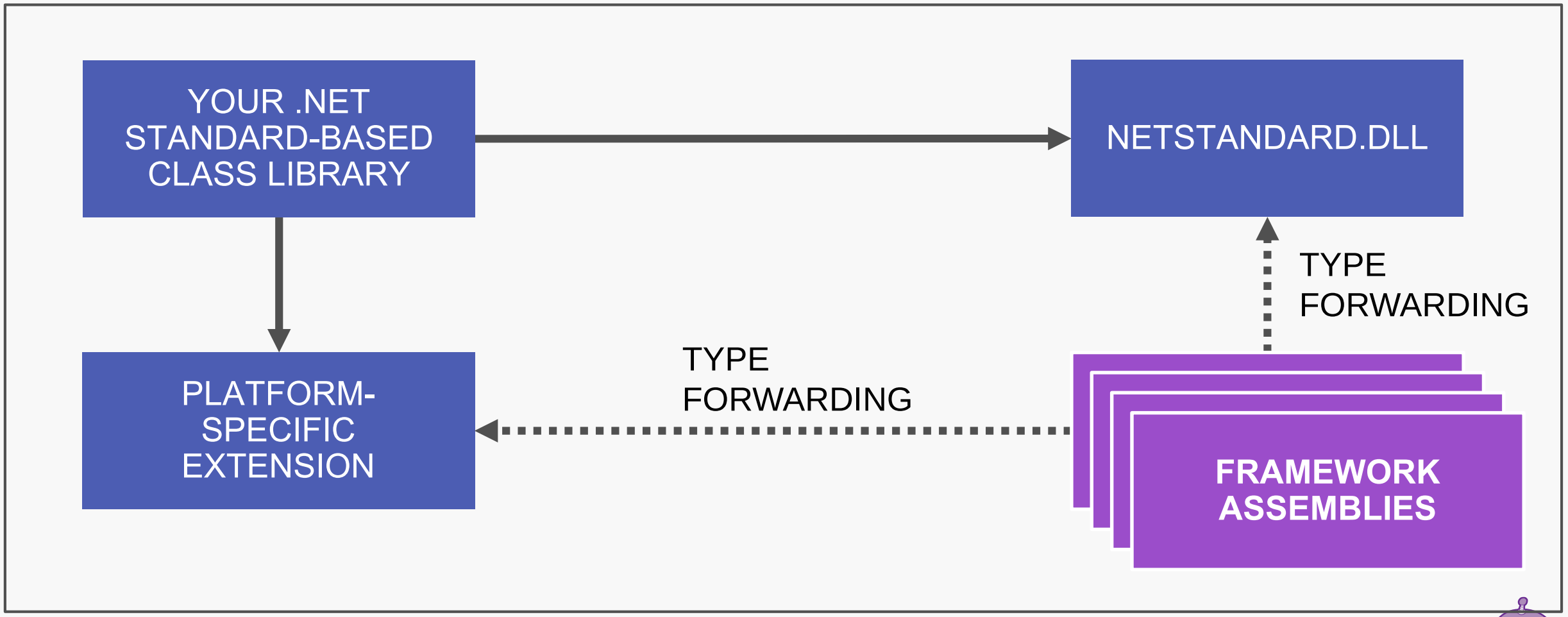
This happens when you build a .NET Standard-based Library

.NET Standard under the hood



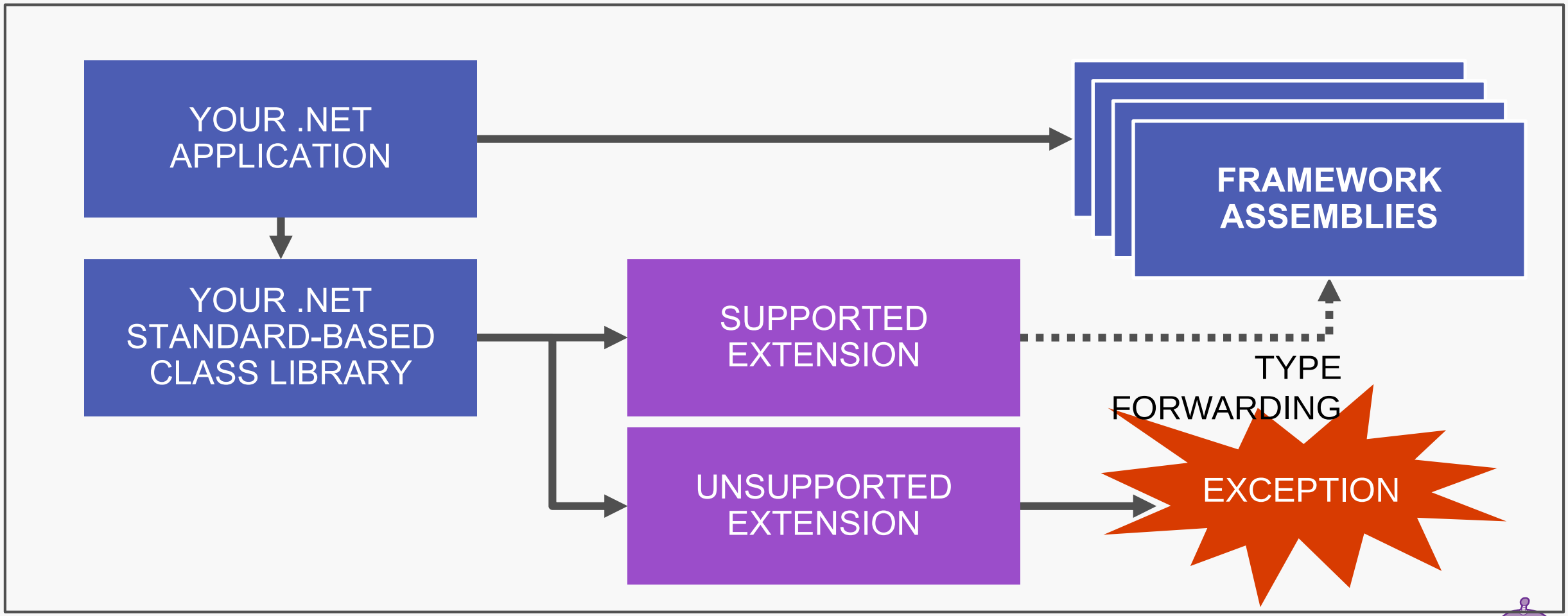
This happens when you load .NET Standard-based library

Platform specific APIs & .NET Standard



This happens when you build a .NET Standard-based library with platform-specific extensions

Platform specific APIs & .NET Standard



This happens when you load .NET Standard-based library with platform-specific extensions