AWS re:Invent

DECEMBER 2 – 6, 2024 | LAS VEGAS, NV

DOP323

.NET application modernization to Linux using Amazon Q Developer

Artur Rodrigues

(he/him)
Principal Solutions Architect
Amazon Web Services

Kirk Davis

(he/him)
Principal Solutions Architect
Amazon Web Services

Before you can modernize

You may have to justify the time and expense of modernizing an application that your leaders think “works fine the way it is”

What are some ways you have talked to leadership to propose a modernization project?

Reasons we hear from executives for modernizing

Reduce costs

- Reduce/eliminate Windows and SQL Server licensing costs
- Create elastic architectures that match costs to actual load
- Use ARM64 architectures for better price-performance

Increase pace of innovation

- Decompose monoliths to smaller services/microservices
- Adopt serverless architectures
- Take advantage of new technologies and C# language features
- Automate manual processes

Improve ability to scale

- Scale individual components/services
- Granular scaling with containers/serverless

Attract and retain talent

Modernizing can cut compute costs substantially

Examples of Amazon EC2 hourly pricing for the same instance type/size

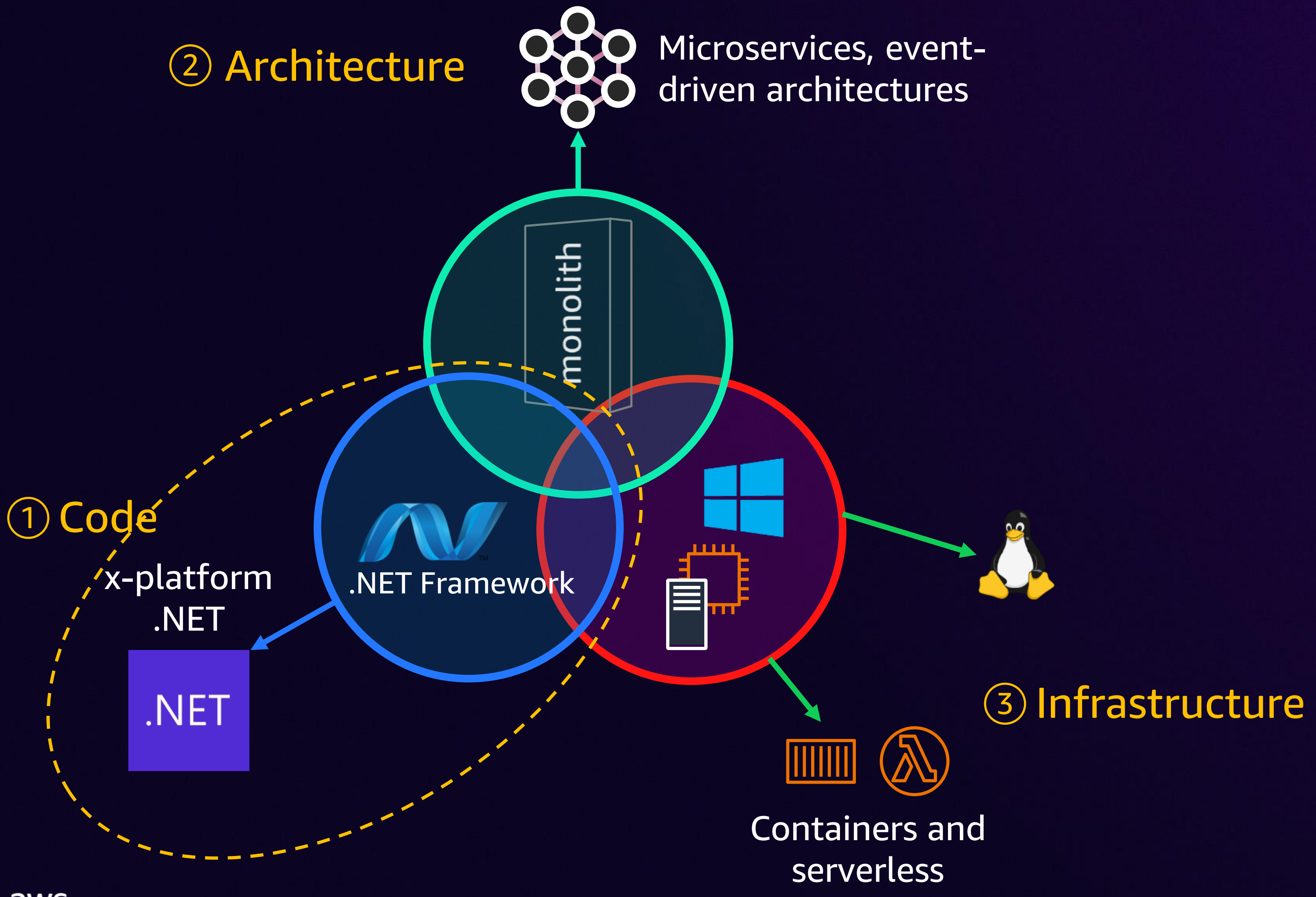
Instance	Windows	Linux	Savings vs. Windows
m7i.2xlarge	$0.7712	$0.4032	**47.7% savings**
c7i.4xlarge	$1.450	$0.714	**50.8% savings**
r7i.8xlarge	$3.5888	$2.1168	**41.0% savings**

Additional savings by moving to Graviton instances

Instance		Linux	vs. Intel+Windows	vs. Intel+Linux
m7g.2xlarge	—	$0.3264	**57.7% savings**	19% savings
c7g.4xlarge	—	$0.5781	**60.1% savings**	19% savings
r8g.8xlarge	—	$1.88512	**47.5% savings**	11% savings

*Ohio Region, July 2024

Modernization can be any/all of the stack



Cross-cutting concerns

- Observability
- CI/CD & DevOps

NET Framework → .NET modernization paths

	.NET Framework	UI conversion	Cross-platform .NET
	Console application	N/A	Console application
	.NET Windows Service	N/A	.NET Worker Service
Web applications	ASP.NET MVC	Minimal to moderate	ASP.NET Core MVC
	ASP.NET Web API	N/A	ASP.NET Core Web API
	ASP.NET Web Forms	Refactor/rewrite	ASP.NET Core MVC, SPA, or potentially CoreWebForms

Issues converting the C# code itself

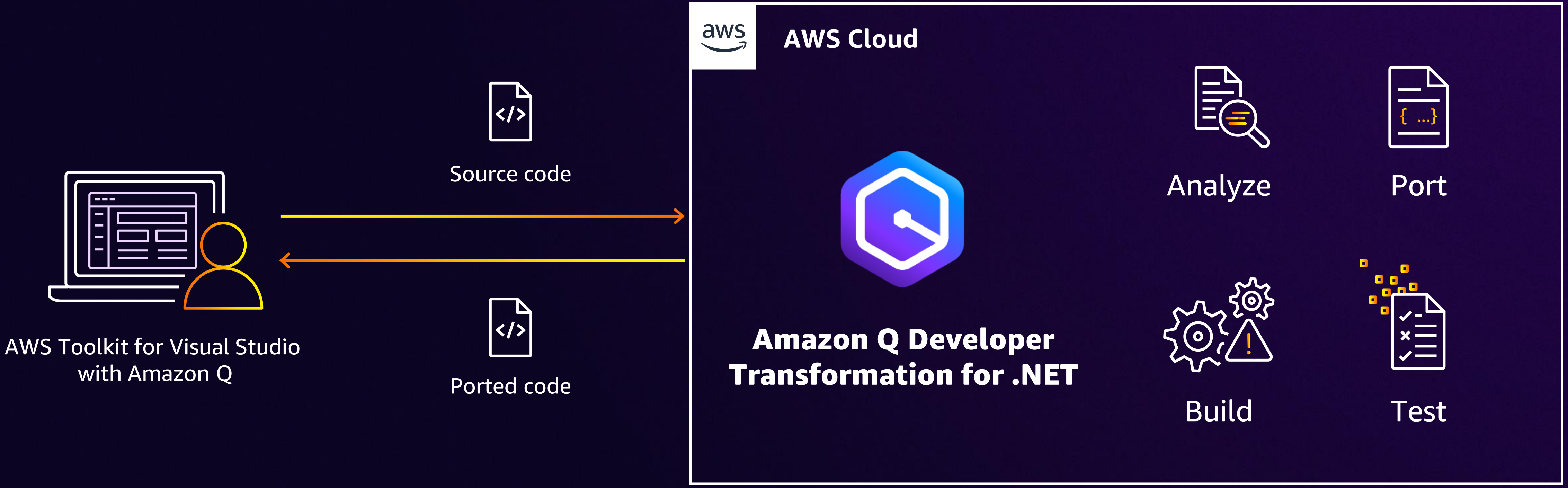
What have been the biggest “code change” roadblocks to porting .NET Framework applications to cross-platform .NET you’ve run into?

Some things we’ve seen customers encounter:

- No HTTP modules or handlers in ASP.NET Core
- HTTP context handled differently
- No EDMX support in EF Core (limited support in EF6)
- App domains, code access security, many System.Web.* classes, and other APIs don’t exist in .NET
- Some APIs moved to different namespaces
- Windows-dependent APIs (original System.Drawing relied on GDI, for example)

Amazon Q Developer Transformation

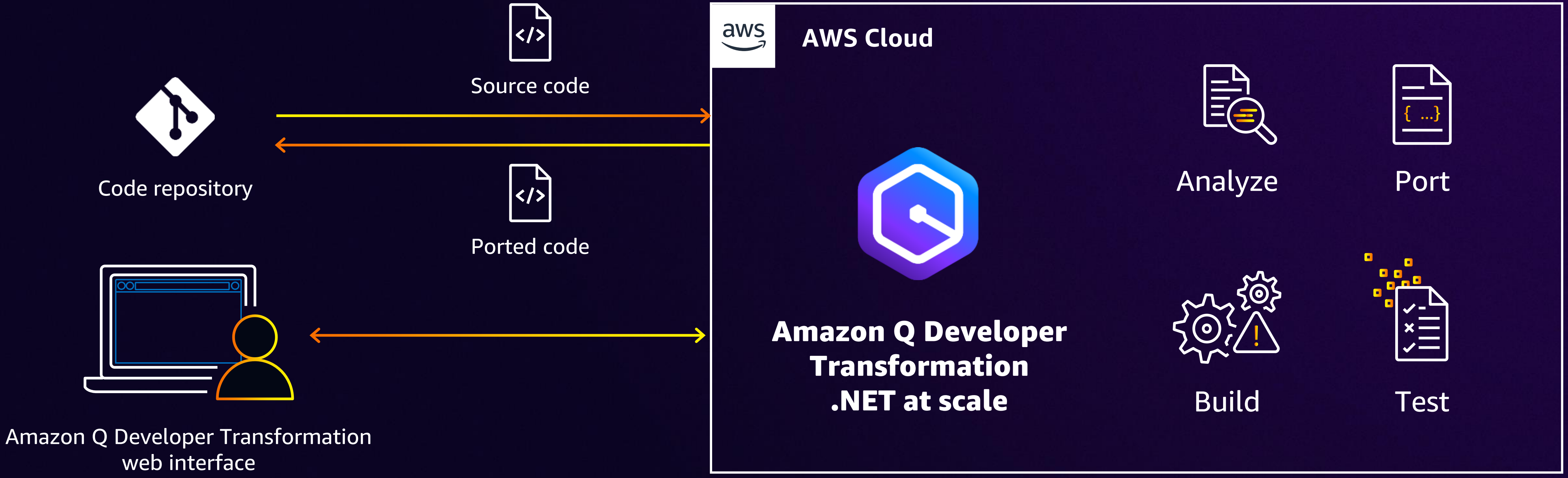
AMAZON Q DEVELOPER TRANSFORMATION FOR .NET IN THE IDE (PREVIEW)


AWS Cloud
Source code
Ported code
AWS Toolkit for Visual Studio with Amazon Q
Amazon Q Developer Transformation for .NET
Analyze
Port
Build
Test

Code transformation demo with Amazon Q Developer Transformation for .NET

Amazon Q Developer Transformation

AMAZON Q DEVELOPER TRANSFORMATION FOR .NET AT SCALE (PREVIEW)



Source code
Code repository
Ported code
Amazon Q Developer Transformation web interface
aws
AWS Cloud
Amazon Q Developer Transformation .NET at scale
Analyze
Port
Build
Test

What the service supports during preview

- .NET projects written in C#
- .NET project types: *
  - Console application
  - Class library
  - WCF service
  - ASP.NET web API
  - Business logic layers of MVC and SPA
- Microsoft-authored NuGet packages
- Default IIS configurations

- Microsoft-authored NuGet packages
- Default IIS configurations
- Source versions:
  - .NET Framework versions 3.5+
  - .NET Core 3.1
  - .NET 5, .NET 7
- Target versions:
  - .NET 6
  - .NET 8
  - .NET 9

* Subject to change

Thank you!

Please complete the session survey in the mobile app

Artur Rodrigues
linkedin.com/in/arturlr

Kirk Davis
linkedin.com/in/kirkdavis