BASTA Spring 2018

C# µServices

Microservices with C#

Rainer Stropek

software architects gmbh

Web	http://www.timecockpit.com
Mail	rainer@timecockpit.com
Twitter	@rstropek

time cockpit

Saves the day.

Your Host

Rainer Stropek

Developer, Entrepreneur
MVP Microsoft Azure
MVP Development Technologies
MS Regional Director
Senior Consultant IT-Visions

Contact

software architects gmbh
rainer@timecockpit.com
Twitter: @rstropek

Introduction

.NET in einer Welt von Microservices

Neues Jahr, neue Vorsätze, neuer C# Workshop. Diesmal setzt Rainer Stropek beim BASTA!-Workshop-Klassiker den Schwerpunkt ganz auf die Entwicklung moderner Microservices-Lösungen mit C# und .NET. An Beispielen zeigt Rainer, wie man mit

- .NET Core,
- den Architekturprinzipien von Microservices
- und der Azure-Cloud

in der Praxis moderne SaaS-Lösungen erstellt.

Natürlich kommen aktuelle Entwicklungen rund um die Sprache C#, .NET und Visual Studio nicht zu kurz. Im C# Workshop sieht man diese Dinge in ein praxisnahes Beispiel verpackt und kann sie so direkt mit in den eignen Arbeitsalltag nehmen.

Microservices

An Introduction

What are Microservices?

Small, autonomous services working together

- Single responsibility principle applied to SOA
- See also concept of Bounded Context

Best used with DevOps and continuous deployment

- Enhance cohesion, decrease coupling, enable incremental evolvement

How small are Microservices?

- It depends (e.g. team structure, DevOps maturity, etc.)
- "... one agile team can build and run it", "... can be rebuilt by a small team in two weeks"
- Find an individual balance

Autonomous = deploy changes without affecting others

- Technology- and platform-agnostic APIs

See also https://en.wikipedia.org/wiki/Microservices

Loose Coupling

Tight Coupling

A change in one module usually forces a ripple effect of changes in other modules

See also Disadvantages of Tight Coupling

Loose Coupling

Components have little or no knowledge of the definitions of other components

Coupling is reduced by e.g. standards, queues, etc.

Microservices = loose coupling wanted

Single change → single deployment

No timing issues (if system A is deployed, system B needs update at the same time)

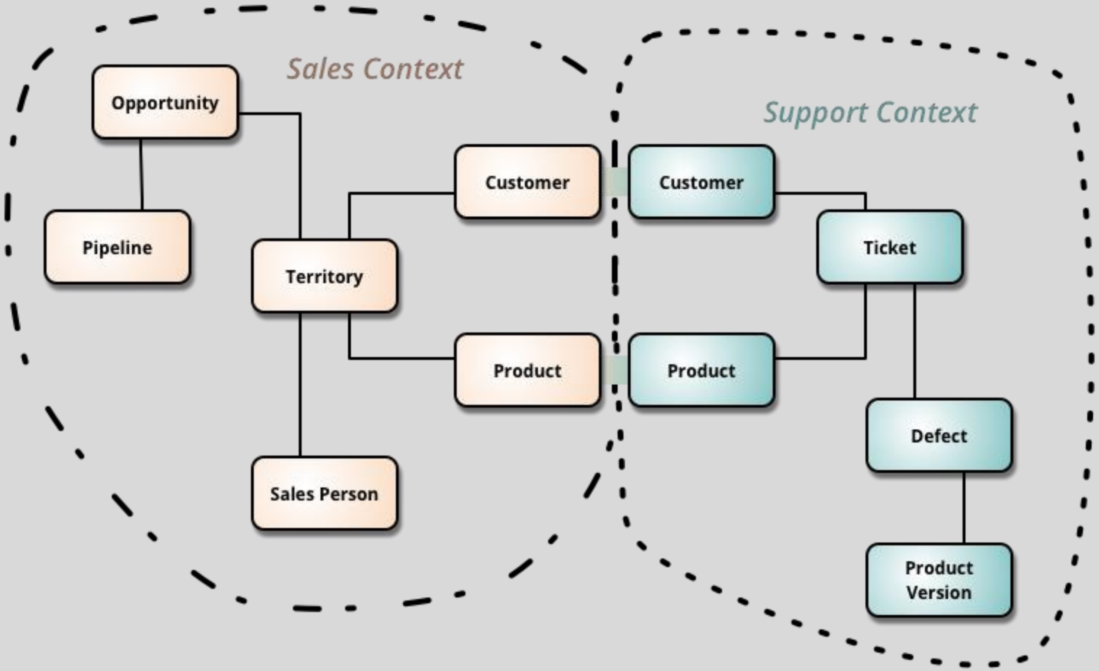
Cohesion

Highly cohesive systems

- Functionality is strongly related
- Modules belong together

Microservices = high cohesion wanted

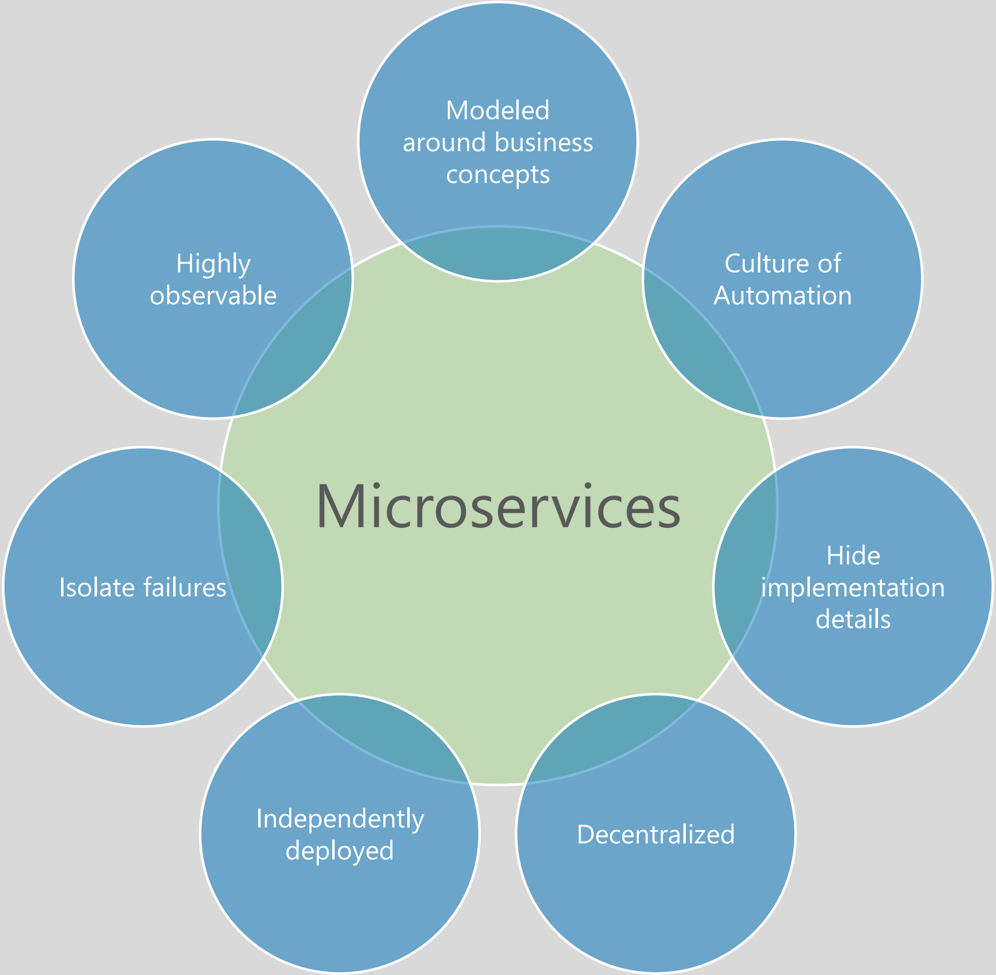
- Functions grouped in a services because all contribute to a single well-defined task
- Reduce risk that a requirement concerns many different system components



Bounded Context

- Microservices often represent bounded contexts
- Business-focused design
  - Less technical-focused design based on technical layers

Source: http://martinfowler.com/bliki/BoundedContext.html



Microservices

Fundamental ideas

- Work alongside many state-of-the-art approaches for software development
  - Agile development techniques
  - Continuous Integration/Delivery
  - DevOps
  - Cloud Computing
  - Containers

Why? Why not?

Why Microservices?

Work well in heterogeneous environments

Right tool for the job
Available skills of team members
Grown environment (e.g. M&A, changing policies, changing overall designs)

Easier to test/adopt new technologies

Reduce risk and cost of failure
New platforms (e.g. Node.js instead of .NET), new versions (e.g. .NET Core),

Resilience

Reduce single point of failures
Support different SLAs for difference modules (costs, agility)
Separation of services add complexity (e.g. network) → criticism of Micrservices

Why Microservices?

Let people take responsibility

Teams “own” their services

You build it, you run it

Scaling

Fine-grained scaling is possible

Simplify deployment of services

Overall, deployment of many Microservices might be more complex → criticism

Deployment patterns: https://www.nginx.com/blog/deploying-microservices/

Why Microservices?

Composability

- Hexagonal architecture

Ability to replace system components

- Outdated technology
- Changed business requirements

Why Not? (Examples)

Harder to debug and troubleshoot

Distributed system

Possible mitigation: Mature logging and telemetry system

Performance penalty

Network calls are relatively slow

Possible mitigation: Remote calls for larger units of work instead of chatty protocols

No strong consistency

We are going to miss transactions!

Possible mitigation: Idempotent retries

Why Not? (Examples)

Harder to manage

- You have to manage lots of services which are redeployed regularly
- Possible mitigation: DevOps, Automation

System is too small

- For small systems, monolithic approach is often more productive
- Cannot manage a monolith (e.g. deployment)? You will have troubles with Microservices!

Environment with lots of restrictions

- Microservices need a high level of autonomy

Team Organization

Conway‘s Law

„Any organization that designs a system will inevitably produce a design whose structure is a copy of the organization's communication structure"

Organizational hurdles for Microservices

- Tightly-coupled organizations
- Geographically distributed teams
- Missing tools (e.g. self-service cloud infrastructure, CI/CD tools)
- Unstable or immature service that frequently changes
- Missing culture of taking ownership (need someone to blame)
- Cope with many different and new technologies

Source: Conway, How Do Committees Invent, Datamation magazine, April 1968

Organisational Helpers

Co-locate teams

- One team responsible for a single service should be co-located

Embrace open source development style

- Works internally, too

Internal consultants, custodians and trusted committers

- Quality gateways
- Servant leaders

Step-by-step approach

Be clear in communication

- E.g. responsibilities, long-term goals, changing roles

Microservices Architects…

…don‘t create perfect end products

…help creating “a framework in which the right systems can emerge, and continue to grow”

…understand the consequences of their decisions

…code with the team (“architects should code”, “coding architect”)

…aims for a balance between standardization and freedom

Build skills for a certain technology vs. right tool for the right job

…create guiding principals and practices

Example for principals (largely technology-independent): https://12factor.net/
Example for practices (often technology-dependent): .NET Core Coding Guildelines

Recommended reading: Newman, Sam. Building Microservices, O'Reilly Media

Guidance, Governance

Samples

Small code samples vs. *perfect* examples from real world

Templates, code generators

Examples: Visual Studio Templates, .NET Core CLI, Angular CLI

Shared libraries

Be careful about tight coupling!

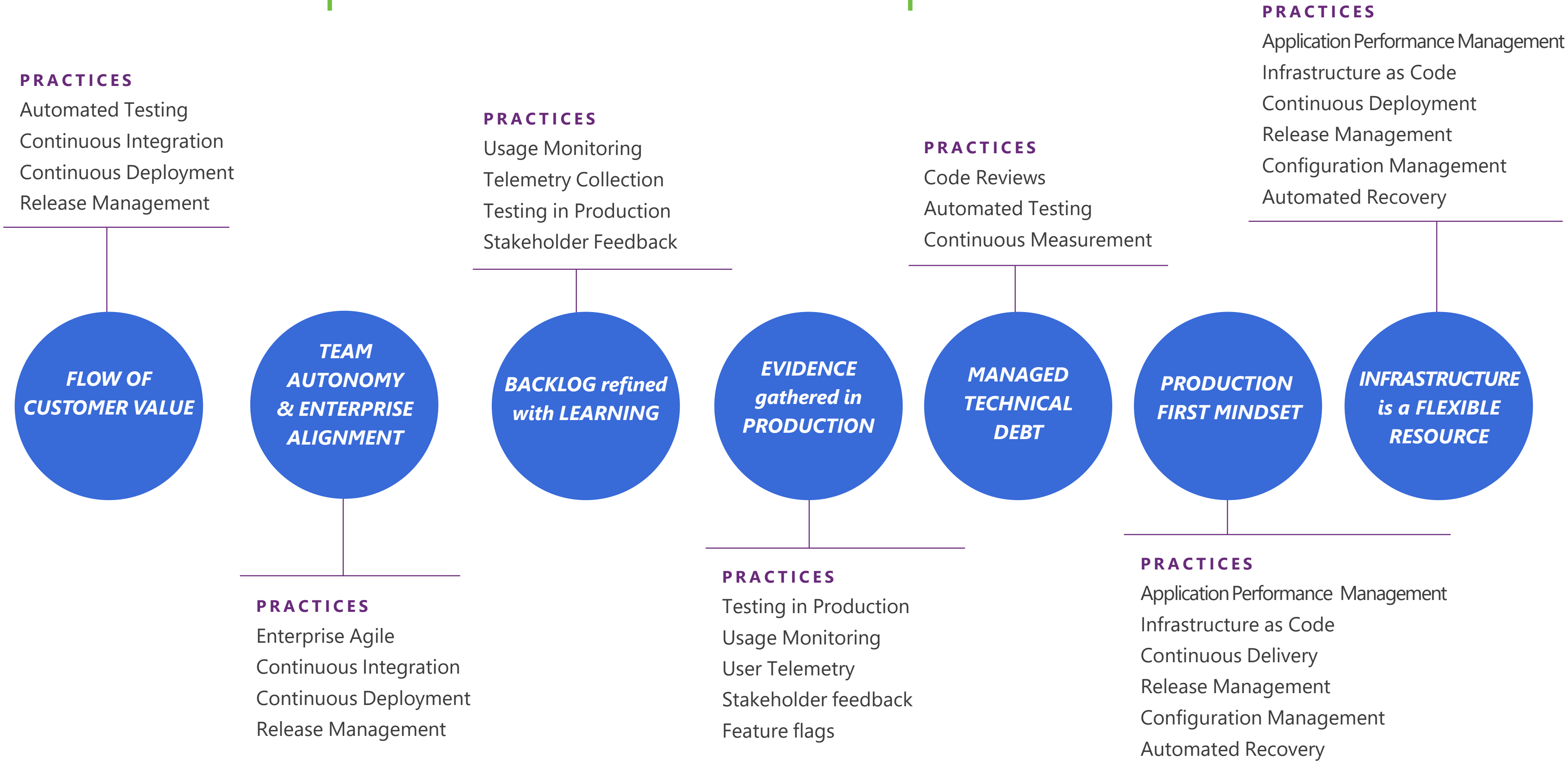
Example: Cross-platform libraries based on .NET Standard Library for cross-cutting concerns

Handle and track exceptions from principals and practices

Remember goal of Microservices: Optimize autonomy

→ Exceptions should be allowed

DevOps habits and practices



FLOW OF CUSTOMER VALUE

PRACTICES

Automated Testing
Continuous Integration
Continuous Deployment
Release Management

TEAM AUTONOMY & ENTERPRISE ALIGNMENT

PRACTICES

Enterprise Agile
Continuous Integration
Continuous Deployment
Release Management

BACKLOG refined with LEARNING

PRACTICES

Usage Monitoring
Telemetry Collection
Testing in Production
Stakeholder Feedback

EVIDENCE gathered in PRODUCTION

PRACTICES

Testing in Production
Usage Monitoring
User Telemetry
Stakeholder feedback
Feature flags

MANAGED TECHNICAL DEBT

PRACTICES

Code Reviews
Automated Testing
Continuous Measurement

PRODUCTION FIRST MINDSET

PRACTICES

Application Performance Management
Infrastructure as Code
Continuous Delivery
Release Management
Configuration Management
Automated Recovery

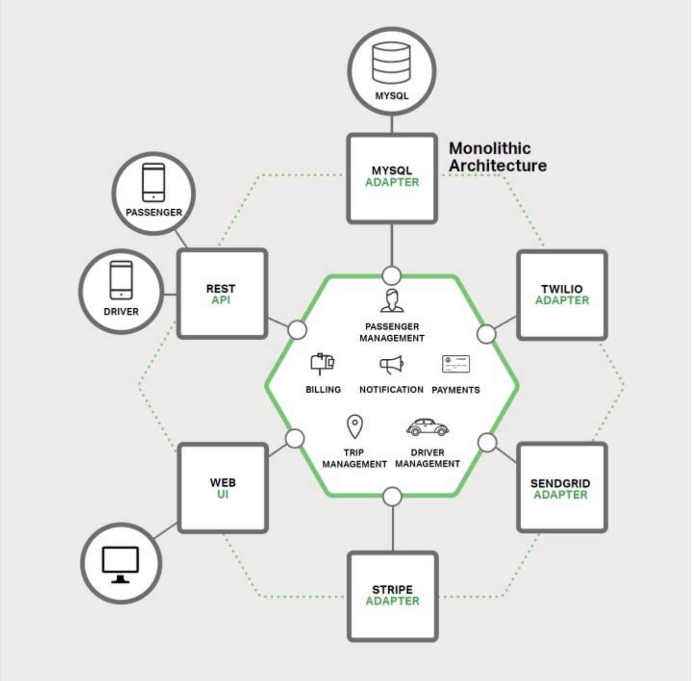
INFRASTRUCTURE is a FLEXIBLE RESOURCE

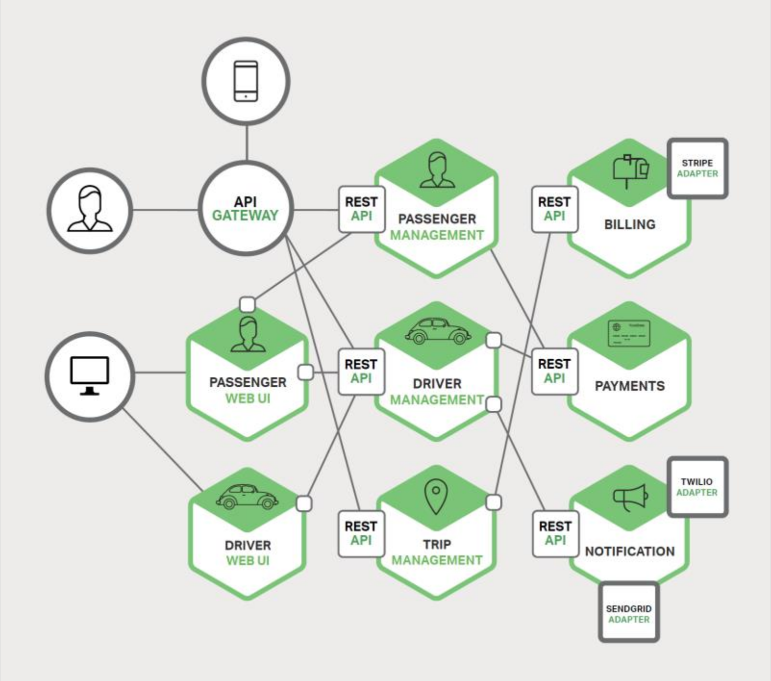
PRACTICES

Application Performance Management
Infrastructure as Code
Continuous Deployment
Release Management
Configuration Management
Automated Recovery

Technical Aspects

Microservice Interfaces





From Monolith to Microservices

Image Source: Chris Richardson, Microservices – From Design to Deployment, NGINX, 2016

Interfaces

Small number of communication standards

- Examples: HTTP/REST, OData, GraphQL, OpenID Connect
- Goals: Interoperability, productivity (economy of scope), detect malfunctions

Practices and principles for typical use-cases needed

- Status Codes
- Data encoding
- Paging
- Dynamic filtering
- Sorting
- Long-running operations
- …

See also https://speakerdeck.com/rstropek/restful-web-api-design

Interface Technology

Tolerant against changes

- See also Breaking Change in Microsoft's REST API Guidelines

Technology-agnostic

Simple to use and provide

- Availability of tools, libraries, frameworks, knowledge

Hide implementation details

- Shared Database anti-pattern

Interface Design

Synchronous communication

Request/response pattern

Bidirectional communication

Example: RESTful Web API, WebSockets

Asynchronous communication

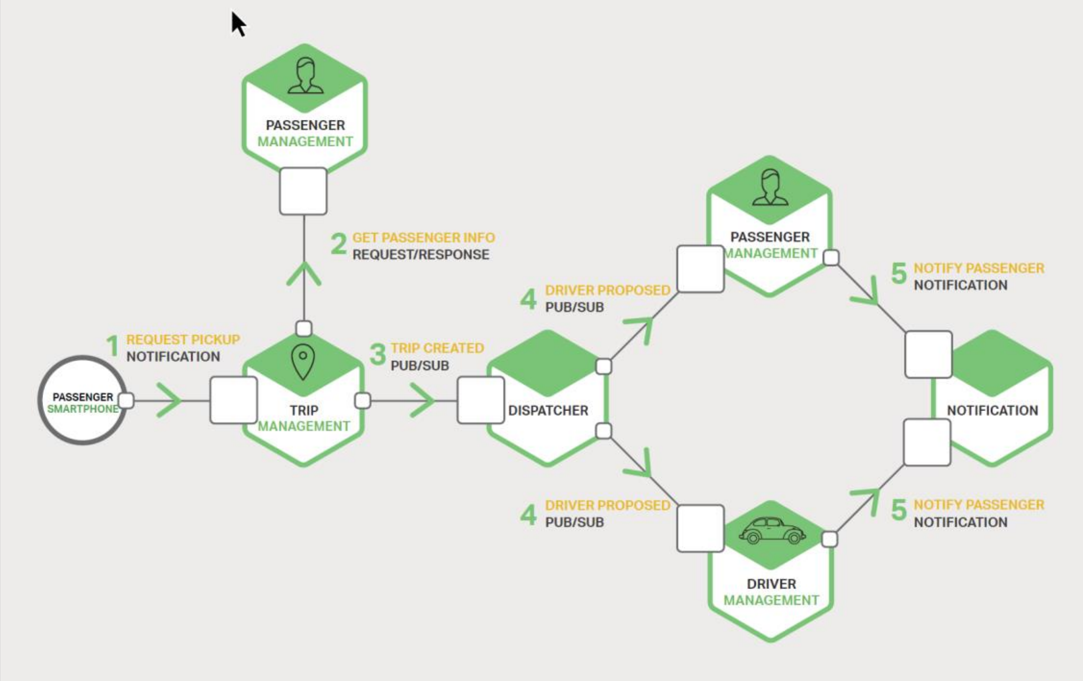
Event-driven

Examples: Service Bus, RabbitMQ, Apache Kafka, Webhooks

Central orchestration or autonomy?

Example: Business Process Modelling and Execution

Further reading: https://www.nginx.com/blog/event-driven-data-management-microservices/


PASSENGER MANAGEMENT
2 GET PASSENGER INFO REQUEST/RESPONSE
1 REQUEST PICKUP NOTIFICATION
PASSENGER SMARTPHONE
TRIP MANAGEMENT
3 TRIP CREATED PUB/SUB
DISPATCHER
4 DRIVER PROPOSED PUB/SUB
PASSENGER MANAGEMENT
5 NOTIFY PASSENGER NOTIFICATION
NOTIFICATION
4 DRIVER PROPOSED PUB/SUB
DRIVER MANAGEMENT
5 NOTIFY PASSENGER NOTIFICATION

Interface Mechanisms

Image Source: Chris Richardson, Microservices – From Design to Deployment, NGINX, 2016

Handling Failures

Partial failures

- Single service must not kill entire system

Outage vs. degradation

- Performance degradation
- Single dependent service not available

Circuit breaker pattern

- Track success of requests
- Stop trying if error rate/performance exceeds threshold
- Regular health check or retry

Versioning

Semantic Versioning (SemVer)

Raise awareness for breaking changes

- Definition of a breaking change is necessary

Avoid breaking changes

- Discussion point: JSON vs. XAML deserializer in C#

Offer multiple versions in parallel

- Give consumers time to move
- Use telemetry to identify slow movers

Libraries vs. Microservices

Goal: Don‘t Repeat Yourself (DRY)

Contraction to Microservices architecture?

Good for…

…cross-cutting concerns (use existing, wide-spread libraries)

…sharing code inside a service boundary

Client libraries

Hide complexity of communication protocol

Implement best practices (e.g. retry policy)

Example: Azure Active Directory Authentication Libraries

UI components

Service provides UI fragments (e.g. WebComponents)

Automation

Continuous Integration and Deployment, Tests

CI/CD

One code repository and CI/CD build per service

- Possible: Common infrastructure for economy of scope and scale

Build and deployment pipeline

- Compile and fast tests (unit tests)
- Slow tests
- UAT (manual tests, explorative tests)
- Performance testing (e.g. cloud load testing)

Separate deployment from release

- E.g. Azure App Service stages with swapping

Canary releasing

- Direct portion of your traffic against new release and monitor stability

Monitoring

System-wide view of our system‘s health

- Contains data from all services
- Logging
- Telemetry (e.g. CPU and memory usage, response times, dependent requests, etc.)

Microsoft‘s solutions

- Visual Studio Application Insights

3rd party solutions

- Log analysis with Elastic Stack
- Dynatrace (leader in Gartner Magic Quadrant)

Manual Testing

Manual testing: try the program and see if it works!

Tester plays the role of a user

- Checks to see if there is any unexpected or undesirable behavior

Test plans with specified test cases

Drawbacks

- Slow
- Requires lots of resources → expensive
- Cannot be performed frequently

Heavy manual testing is a showstopper for Microservices

Testing Level

Unit Test

Test single function or class

Service Tests

Bypass UI and test service directly

Stubs or mockups for dependent services/resources (e.g. Mountbank)

End-to-End Tests

Hard in a Microservice environment (e.g. which versions to test?)

Tend to be flaky and brittle

Good approach: Test a few customer-driven "journeys"

Deployment

Deployment Strategies

Single service instance per host

- Inefficient

Multiple service instances per host

- Efficient in terms of resource usage
- No isolation → no resource limitation, no isolated environments, no sandboxes

Service instance per VM

- Based on a common image
- Complete isolation
- Uses resources less efficient → expensive
- Requires mature virtualization environment

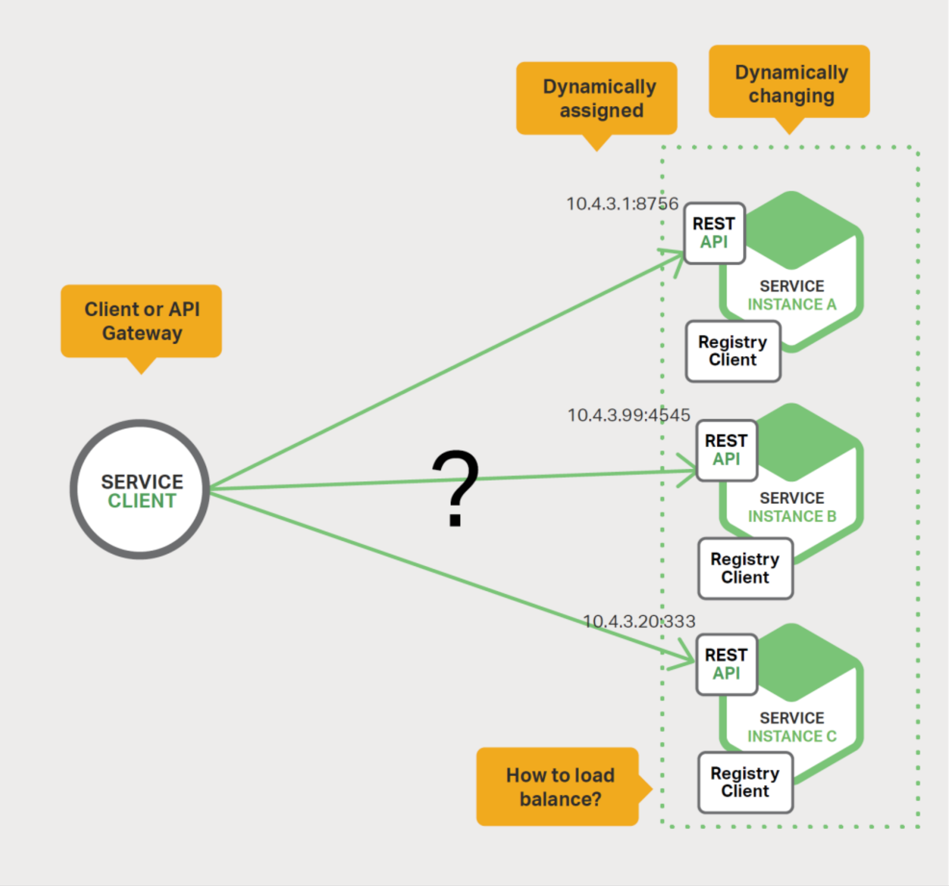
Deployment Strategies

Service instance per container

- Based on a common image (automatically created)
- High level of isolation (like VMs if you use e.g. Windows Hyper-V Container)
- Requires running container environment (e.g. Docker Cloud, Azure Container Services)

Serverless deployments

- E.g. Azure App Service, Azure Functions
- Reduce operations to a minimum



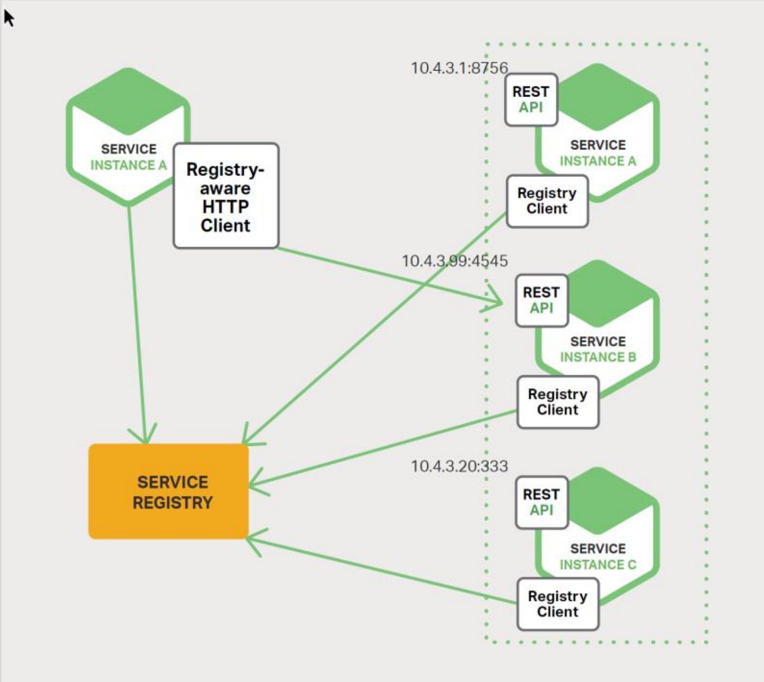
Service Discovery

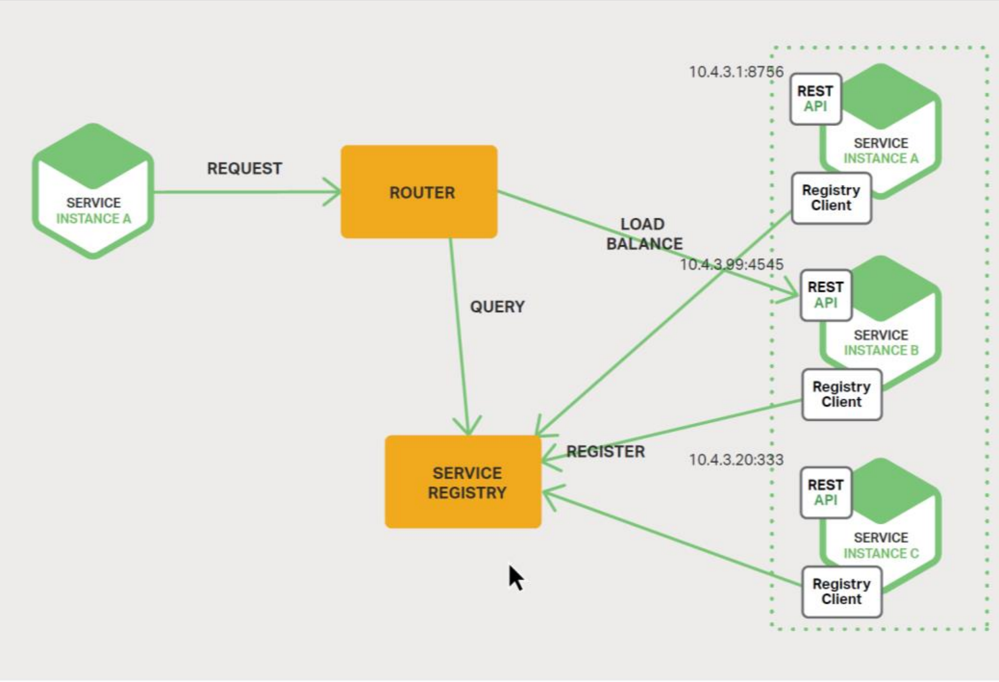
Dynamically assigned addresses

Changing environment
- Failures
- Scaling
- New versions

Tools
- DNS (e.g. Azure DNS)
- Load Balancer (e.g. Azure LB)
- Discovery and config tools (e.g. Consul)

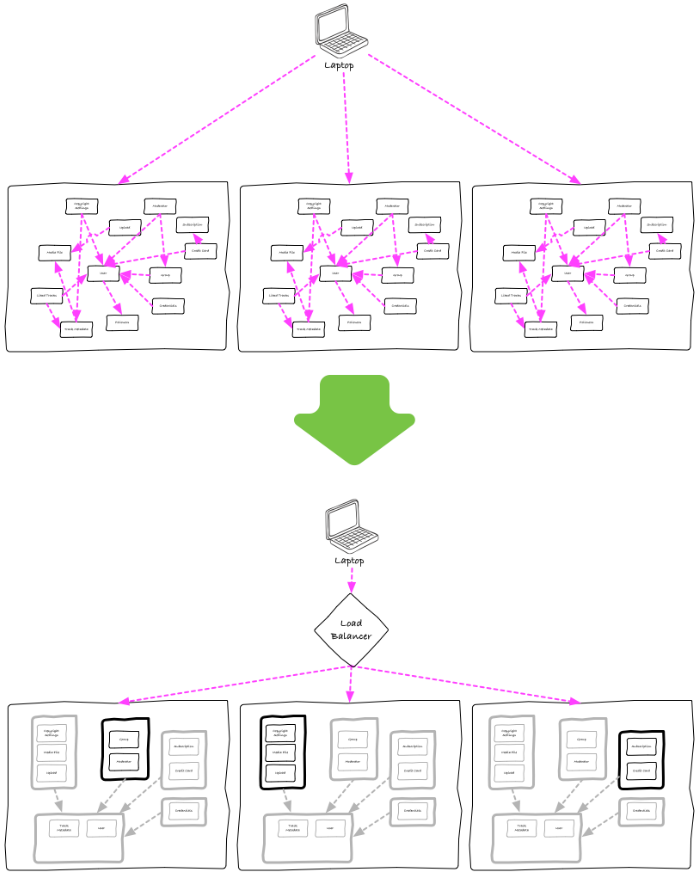
Image Source: Chris Richardson, Microservices – From Design to Deployment, NGINX, 2016





Client vs. server-side discovery

Image Source: Chris Richardson, Microservices – From Design to Deployment, NGINX, 2016



Deployment

Architecture

- Old: Each node contains entire system
- New: Unrelated modules behind load balancer/reverse proxy

API Gateways

- Marshal backend calls
- Aggregate content
- Example: [Azure API Management]()

Source: [How we ended up with microservices]()

Data Management

Data Management

Each Microservice has its own data

- No transactions
- No distributed queries
- Duplicated data to a certain extent

Event-driven architecture

- Requires service bus or message broker (e.g. Service Bus, RabbitMQ, Apache Kafka)
- Option: Use DB transaction log

Event sourcing and CQRS

- Read more in MSDN, Martin Fowler

Transactions

Question and avoid ACID transactions across services

- Perfectly fine inside service boundaries
- Has consequences on API design (e.g. Azure Storage Entity Group Transactions)

Idempotent retry

- Gather data, try again later

Use compensating transactions

Further Readings

Further Readings

Martin Fowler on Microservices

Newman, Sam. Building Microservices, O'Reilly Media

NGINX

- Tech Blog
- Microservices: From Design to Deployment

.NET Core

Why .NET Core?

Refactor .NET Framework

Establish a Standard Library for the various incarnations of .NET

.NET Core is not 100% compatible with .NET 4.x (details)

Make it a real cross-platform solution

Windows, Mac OS, Linux (details in .NET Core Roadmap)

Make it open source

A .NET Foundation project

MIT License

Details: https://docs.microsoft.com/dotnet/

Components of .NET Core

.NET Runtime (CoreCLR)

- CoreCLR includes Base Class Library (BCL)

.NET Core Foundation Libraries (CoreFX)

.NET Command Line Tools (.NET CLI)

- Including the **`dotnet`** application host

Cross-Platform Compiler (Roslyn)

Status of .NET Core

.NET Core 2.0 is RTM (Aug. 2017)

- Download current version
- 2.1 is scheduled for 2018 (roadmap)

Visual Studio Tools are RTM

- Visual Studio 2017

C# is RTM

X64 Support

- X86, X64 support on Windows
- X64 support on many Linux distros
- Community-supported version for Raspberry Pi

See also: https://github.com/dotnet/core/blob/master/roadmap.md

What can you build?

Console applications

ASP.NET Core applications

UWP applications

Xamarin Forms applications

See also: https://github.com/dotnet/core/blob/master/roadmap.md

Where to get .NET Core?

.NET Core landing page

With Visual Studio tools (Visual Studio prerequisites)
Command-line tools (with your own editor, e.g. VSCode, download)

.NET Install Script (details, download)

You have to care for the prerequisites

NuGet

Packages and Metapackages

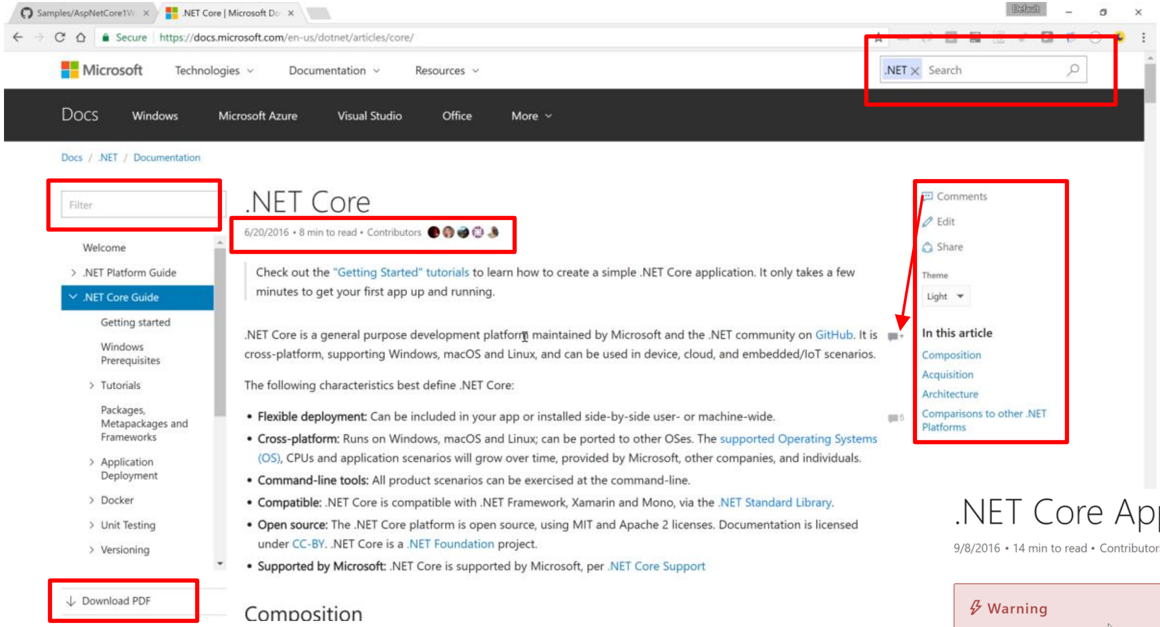
Docker: `microsoft/dotnet` image (details)

.NET Core Source Browser

See also: https://github.com/dotnet/core/blob/master/roadmap.md

Getting Help

New https://docs.microsoft.com



.NET Core

6/20/2016 • 8 min to read • Contributors

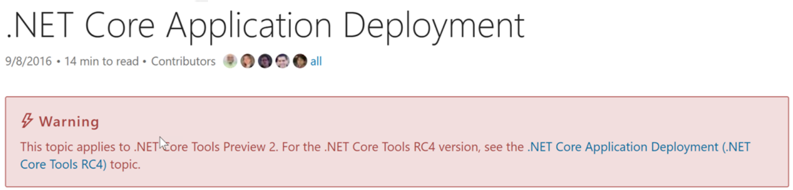
> Check out the "Getting Started" tutorials to learn how to create a simple .NET Core application. It only takes a few minutes to get your first app up and running.

.NET Core is a general purpose development platform maintained by Microsoft and the .NET community on GitHub. It is cross-platform, supporting Windows, macOS and Linux, and can be used in device, cloud, and embedded/IoT scenarios.

The following characteristics best define .NET Core:

- **Flexible deployment:** Can be included in your app or installed side-by-side user- or machine-wide.
- **Cross-platform:** Runs on Windows, macOS and Linux; can be ported to other OSes. The supported Operating Systems (OS), CPUs and application scenarios will grow over time, provided by Microsoft, other companies, and individuals.
- **Command-line tools:** All product scenarios can be exercised at the command-line.
- **Compatible:** .NET Core is compatible with .NET Framework, Xamarin and Mono, via the .NET Standard Library.
- **Open source:** The .NET Core platform is open source, using MIT and Apache 2 licenses. Documentation is licensed under CC-BY. .NET Core is a .NET Foundation project.
- **Supported by Microsoft:** .NET Core is supported by Microsoft, per .NET Core Support

Composition



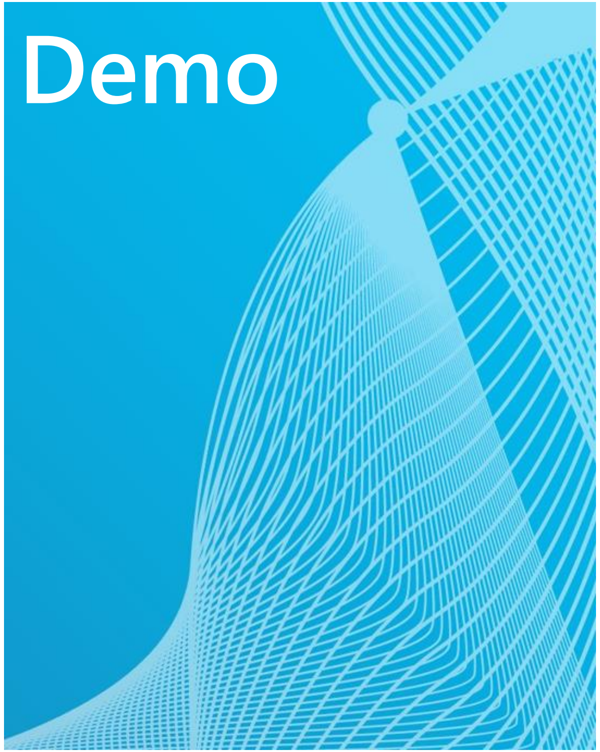
.NET Core Application Deployment

9/8/2016 • 14 min to read • Contributors all

Warning

This topic applies to .NET Core Tools Preview 2. For the .NET Core Tools RC4 version, see the .NET Core Application Deployment (.NET Core Tools RC4) topic.

Packages, Metapackages and Frameworks



Demo

Create console app with CLI

Analyze `.csproj`

Discuss `.csproj` reference

Run app

Publish app

Further readings

- [More about cross-platform libraries]
- [MSBuild Project File Schema Reference]
- [Creating new templates]
- [Runtime Configuration Files]

https://github.com/rstropek/Samples/tree/master/AspNetCore1Workshop/10-console-hello-world

.csproj

```
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>netcoreapp2.0</TargetFramework>
  </PropertyGroup>
</Project>
```

For executable, not present for class libraries

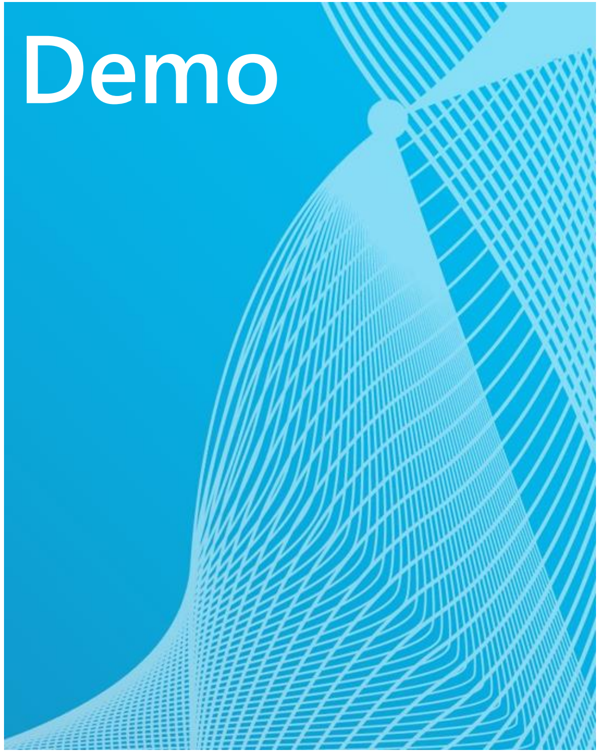
Target Framework

```
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <TargetFramework>netstandard2.0</TargetFramework>
  </PropertyGroup>
</Project>
```

Class library based on .NET Standard 2.0

See also [MSBuild Project File Schema Reference]

Solutions



Create solution: **`dotnet new sln`**

Add proj.: **`dotnet sln add …`**

Create solution in VS2017

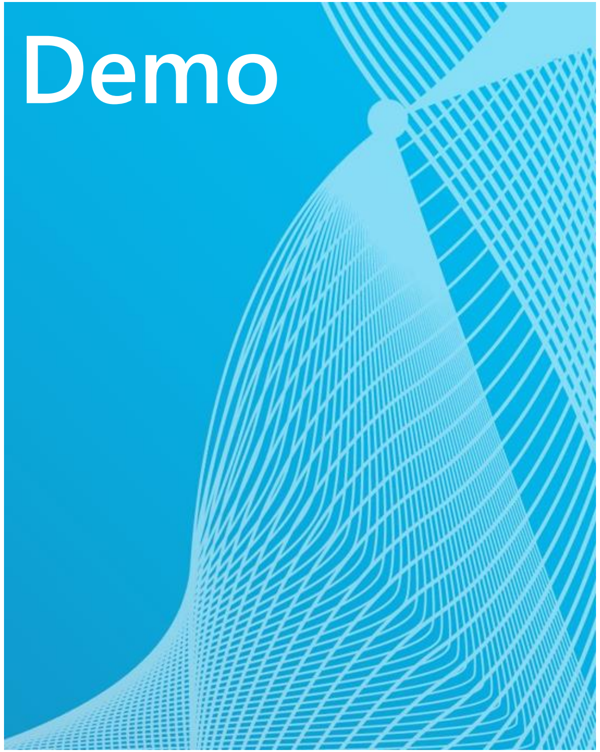
- .NET Standard class library with Json.NET
- .NET Framework console app with reference

Further readings

- [.NET Core Tools MSBuild]()

https://github.com/rstropek/Samples/tree/master/AspNetCore1Workshop/25-project-references

Cross-platform



Run app on Linux using Docker

`microsoft/dotnet` images

Multi-step build

Docker support in VS2017

See also https://github.com/dotnet/dotnet-docker-samples

.NET CLI

.NET Core CLI

`dotnet` command

- `new` – create project
- `migrate` –*project.json* → *.csproj*
- `restore` – restore dependencies
- `run` – run source code without explicit compile
- `build` – builds project and dependencies
- `test` – runs unit tests
- `pack` – packs code into a NuGet package
- `publish` – packs the app and dependencies for publishing

https://docs.microsoft.com/en-us/dotnet/core/tools/index?tabs=netcore2x

`dotnet run`

Run application from the source code

- Use **`dotnet`** without any command to run a built DLL

Uses **`dotnet build`** in the background

Important parameters

- `--framework`
- `--configuration <Debug|Release>`

https://docs.microsoft.com/en-us/dotnet/articles/core/preview3/tools/dotnet-run

Deployment (`dotnet publish`)

Framework-dependent deployment

Shared system-wide version of .NET Core must be present on target system

DLLs are launched using **`dotnet`**

DLLs are portable

Self-contained deployment

No prerequisites on target system necessary

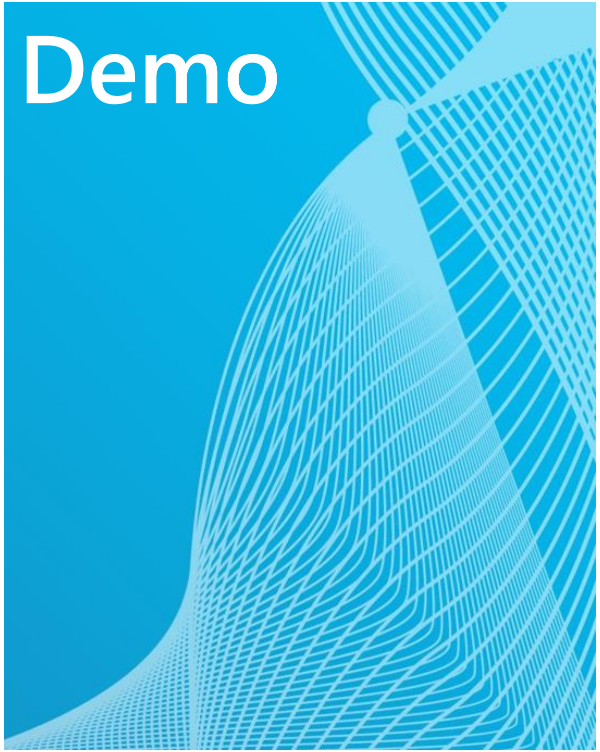
Does *not* contain native prerequisites

Results in an platform-specific executable

Optional: Use CrossGen for native image generation

https://docs.microsoft.com/en-us/dotnet/core/deploying/

Self-contained Deployment



Create self-contained sample

See following slides

Build and publish SCD

```
dotnet publish -c release
dotnet publish -c release -r win-x64
dotnet publish -c release -r linux-x64
```

Runtime Identifier (RID)
(details)

Release instead of debug version
(need not ship PDBs)

https://github.com/rstropek/Samples/tree/master/AspNetCore1Workshop/27-self-contained

Versioning

Versioning

Framework version changes when APIs are added

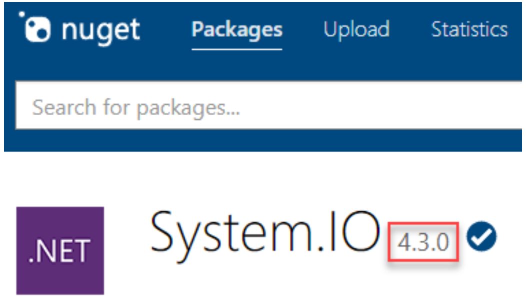
No implementation → no patch numbers

Example: **`netcoreapp2.0`**

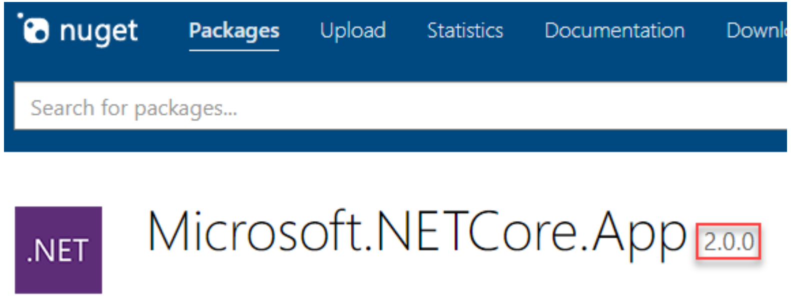
Package versions

`System.*` packages use 4.x numbers (overlap with .NET Framework)

Packages without overlapping with .NET Framework → 1.x/2.x



System.IO 4.3.0



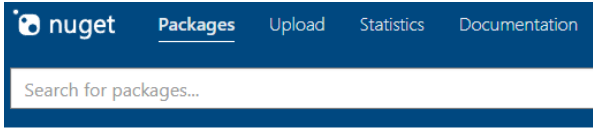
Microsoft.NETCore.App 2.0.0

https://docs.microsoft.com/en-us/dotnet/articles/core/versions/index

Versioning

.NET Standard Library

- Versioning independent of any .NET runtime, applicable to multiple runtimes
- 2.0 for .NET Core 2.0



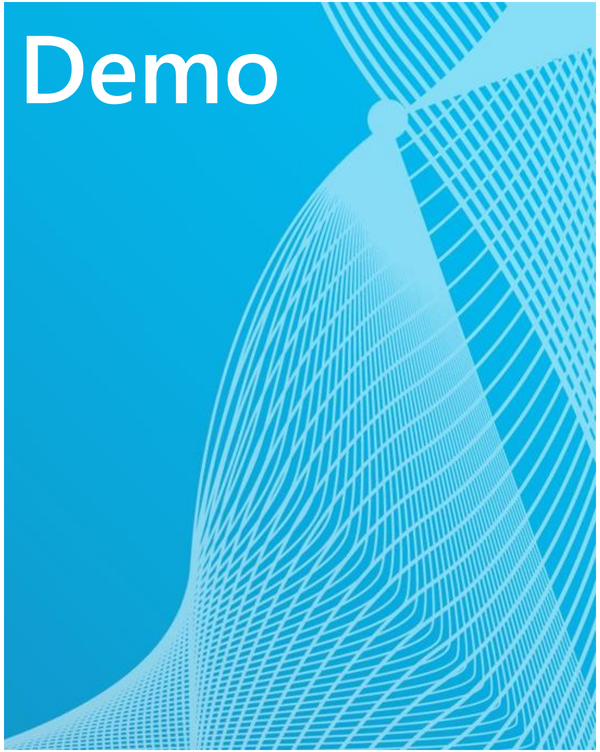
NETStandard.Library 2.0.0

[Examples]

https://docs.microsoft.com/en-us/dotnet/articles/core/versions/index

Libraries

Libraries



Demo

Shared files

Libraries

Creating NuGet packages

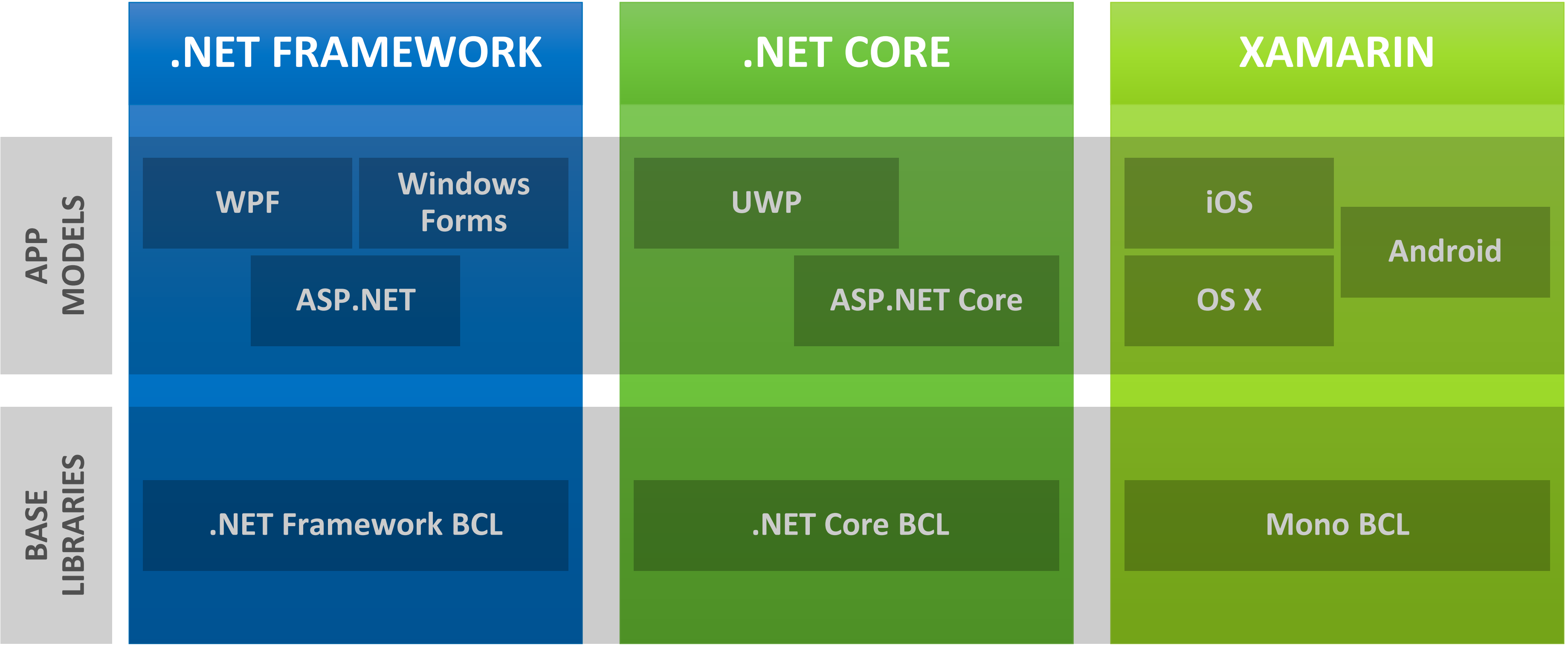
```
dotnet pack
```

Further readings

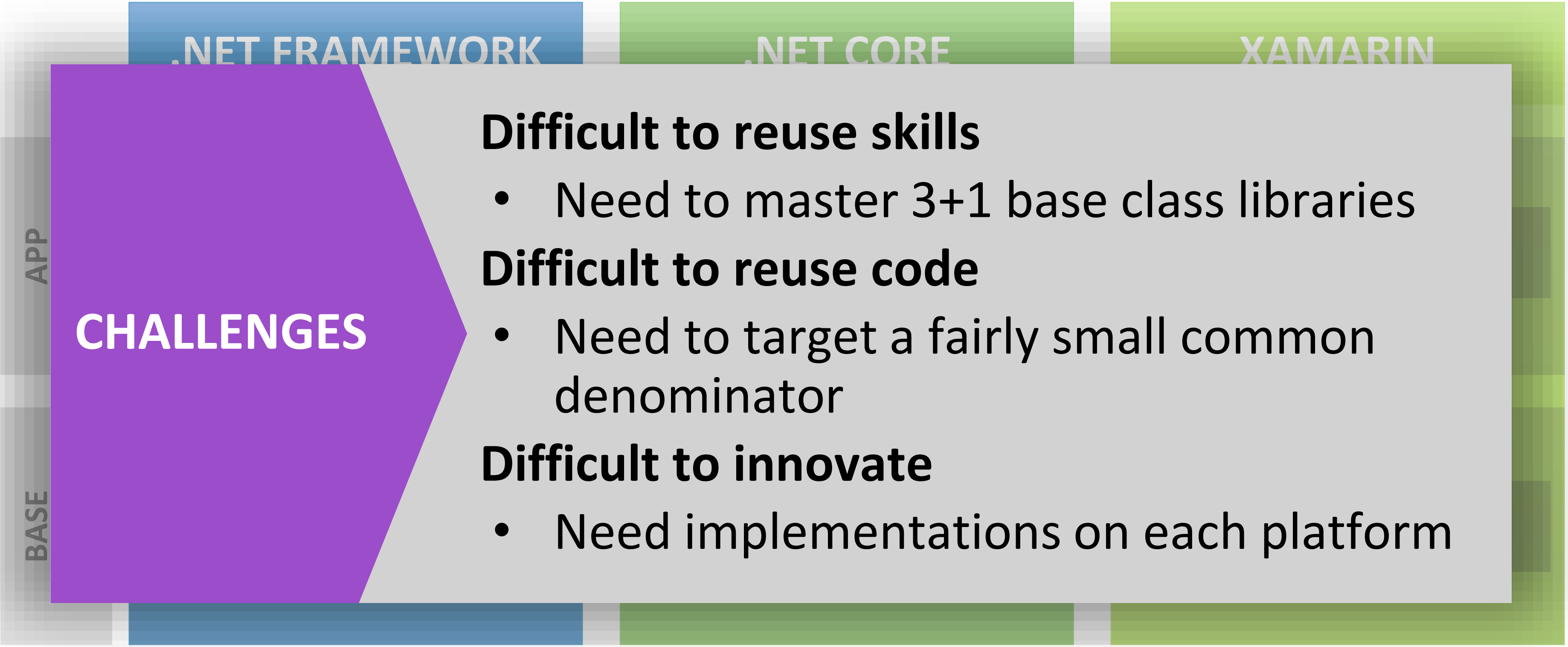
- [More about cross-platform libraries]()
- [Tools for porting code from .NET Framework]()

.NET Standard Library

.NET today—reusing code



	.NET FRAMEWORK	.NET CORE	XAMARIN
APP MODELS	WPF Windows Forms ASP.NET	UWP ASP.NET Core	iOS OS X Android
BASE LIBRARIES	.NET Framework BCL	.NET Core BCL	Mono BCL



CHALLENGES

Difficult to reuse skills

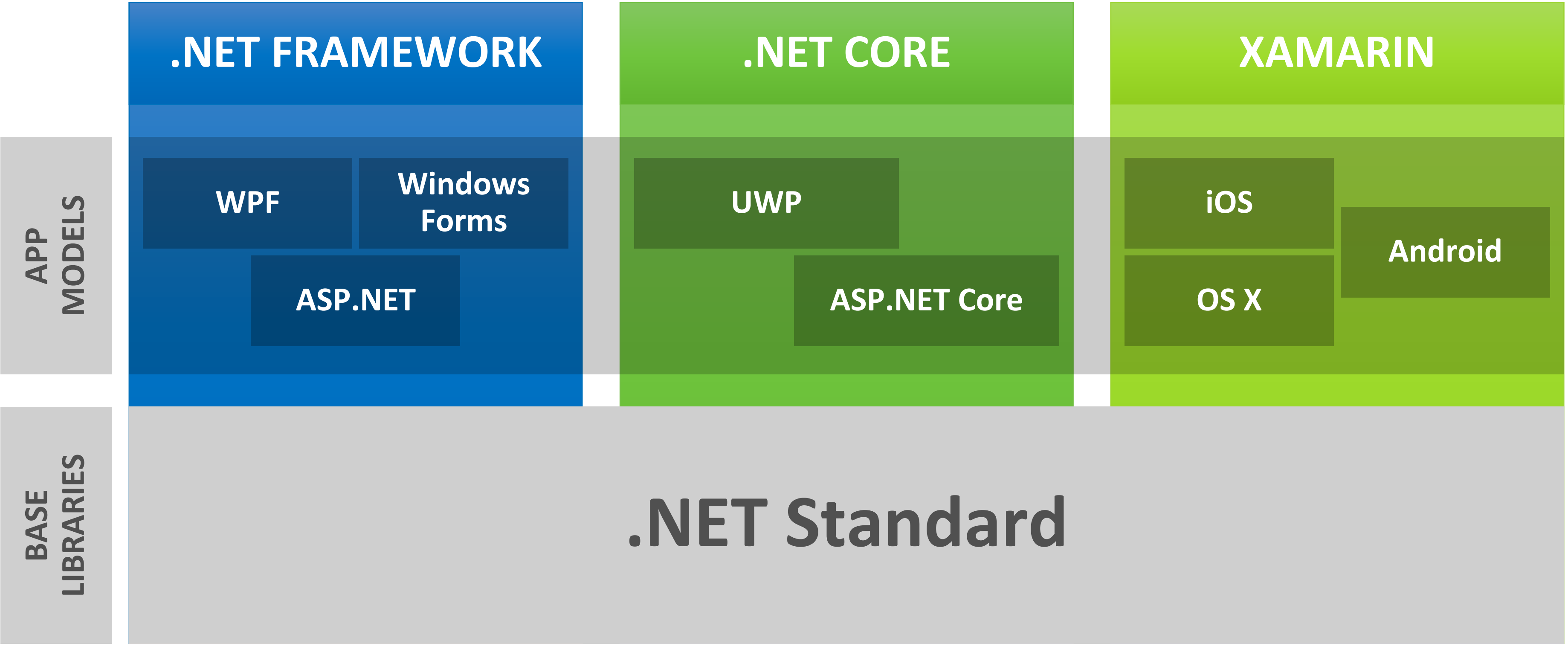
- Need to master 3+1 base class libraries

Difficult to reuse code

- Need to target a fairly small common denominator

Difficult to innovate

- Need implementations on each platform



APP MODELS

BASE LIBRARIES

.NET FRAMEWORK

WPF

Windows Forms

ASP.NET

.NET CORE

UWP

ASP.NET Core

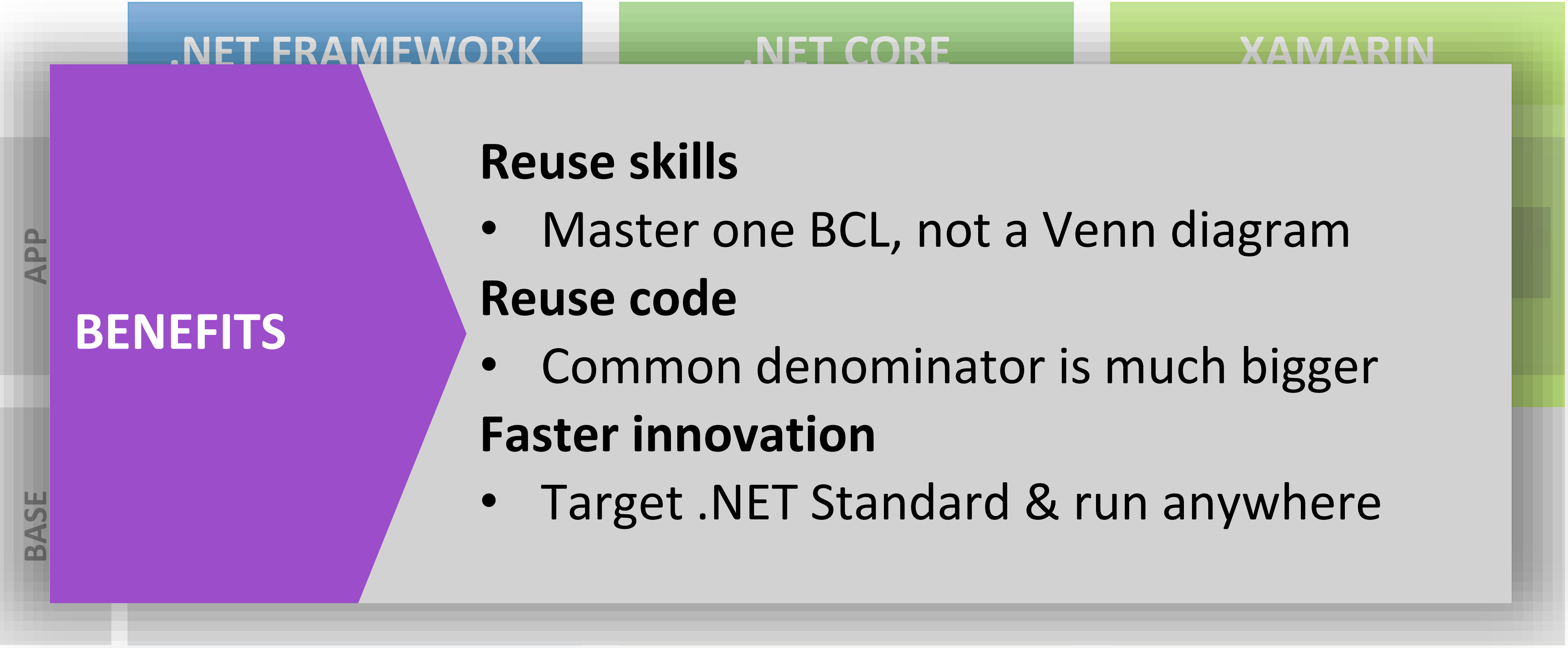
XAMARIN

iOS

OS X

Android

.NET Standard



BENEFITS

Reuse skills

- Master one BCL, not a Venn diagram

Reuse code

- Common denominator is much bigger

Faster innovation

- Target .NET Standard & run anywhere

What is .NET Standard?

.NET Standard **is a specification**

A set of APIs that **all .NET platforms have to implement**

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

.NET Standard 2.0

Has much bigger API surface

Extended to cover intersection between .NET Framework a

Makes .NET Core 2.0 bigger as it implements .NET Standar

+20K

More APIs than
.NET Standard 1.x

Can reference .NET Framework libraries

- Compat shim allows referencing existing .NET Framework code – without recompilation
- Limited to libs that use APIs that are available for .NET Standard

~70%

of NuGet packages
are API compatible

Why a standard library?

CLR (CLI) has already been standardized (ECMA 334)

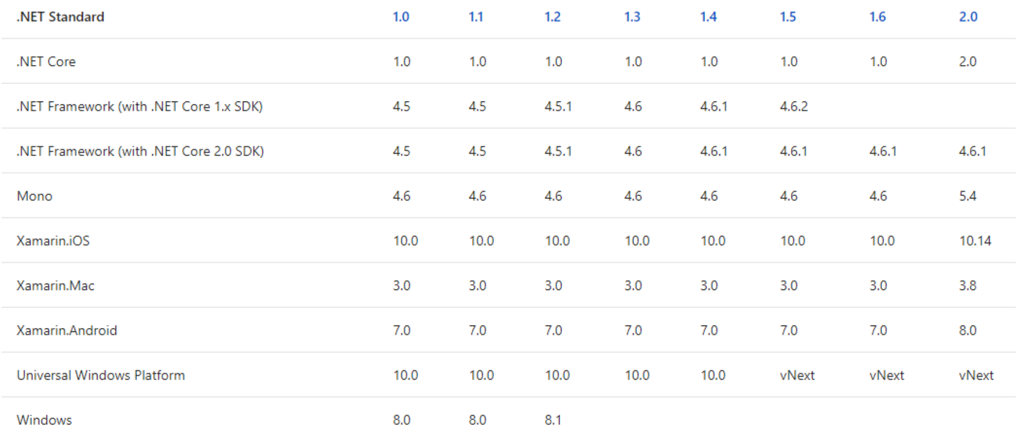
- No standardized BCL prior to .NET Core

Goal: Standard BCL API for all .NET platforms

- Easier to create portable libraries
- Reduce conditional compilation

What about PCLs?

- Well defined API instead of just intersection of platforms
- Better versioning
- Overlapping PCL profiles (details)



.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 (with .NET Core 1.x SDK)	4.5	4.5	4.5.1	4.6	4.6.1	4.6.2		
.NET Framework (with .NET Core 2.0 SDK)	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	4.6	4.6	4.6	5.4
Xamarin.iOS	10.0	10.0	10.0	10.0	10.0	10.0	10.0	10.14
Xamarin.Mac	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.8
Xamarin.Android	7.0	7.0	7.0	7.0	7.0	7.0	7.0	8.0
Universal Windows Platform	10.0	10.0	10.0	10.0	10.0	vNext	vNext	vNext
Windows	8.0	8.0	8.1					

Details: https://docs.microsoft.com/en-us/dotnet/articles/standard/library

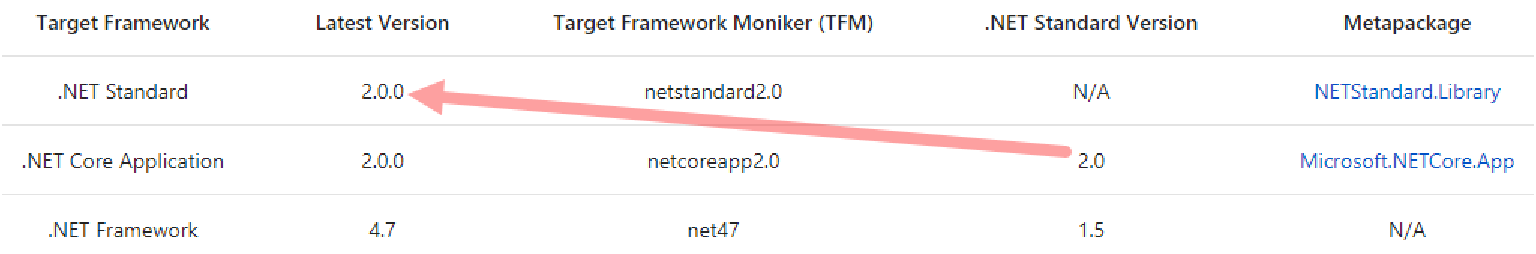
.NET Standard Library

Standard APIs defined as empty C# classes

Example: `ref` folder in `System.Runtime`

`NETStandard.Library` (NuGet)

Metapackage for .NET Standard Library



Target Framework	Latest Version	Target Framework Moniker (TFM)	.NET Standard Version	Metapackage
.NET Standard	2.0.0	netstandard2.0	N/A	NETStandard.Library
.NET Core Application	2.0.0	netcoreapp2.0	2.0	Microsoft.NETCore.App
.NET Framework	4.7	net47	1.5	N/A

Details: https://docs.microsoft.com/en-us/dotnet/articles/standard/frameworks

Migration

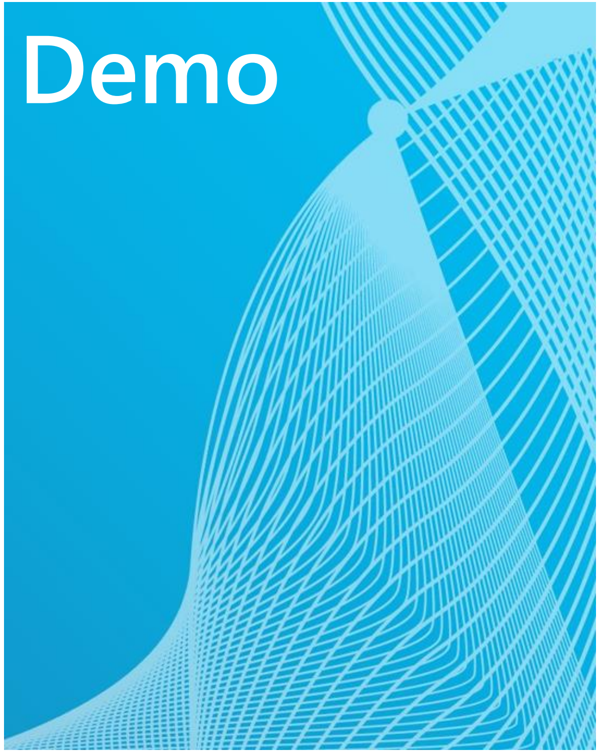
.NET Portability Analyzer

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

Reference .NET Framework assemblies

- They just work, without recompile

.NET Portability Analyzer



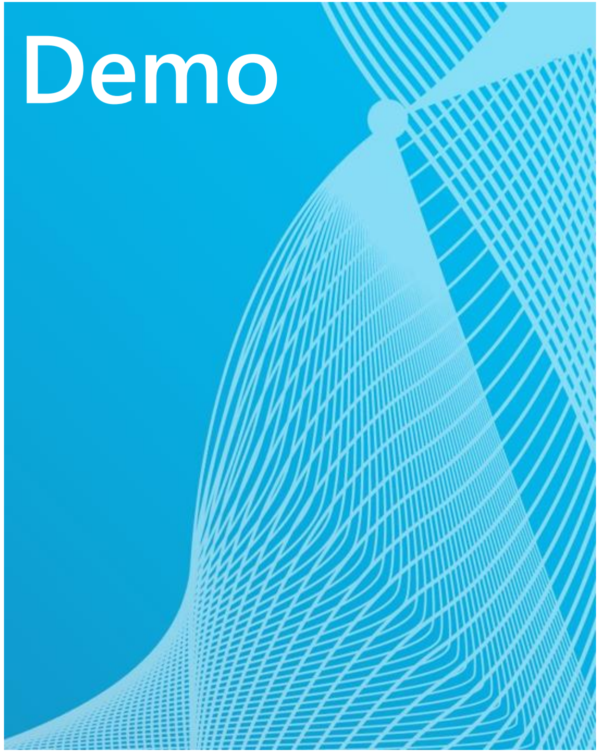
Demo

NQuery

ASP.NET Core Basics

Practical use of .NET Core

Minimal ASP.NET Core



Demo

ASP.NET Pipeline

Discuss "a la carte" framework

- Add static files (sample)

Kestrel

- Windows, Linux with Docker

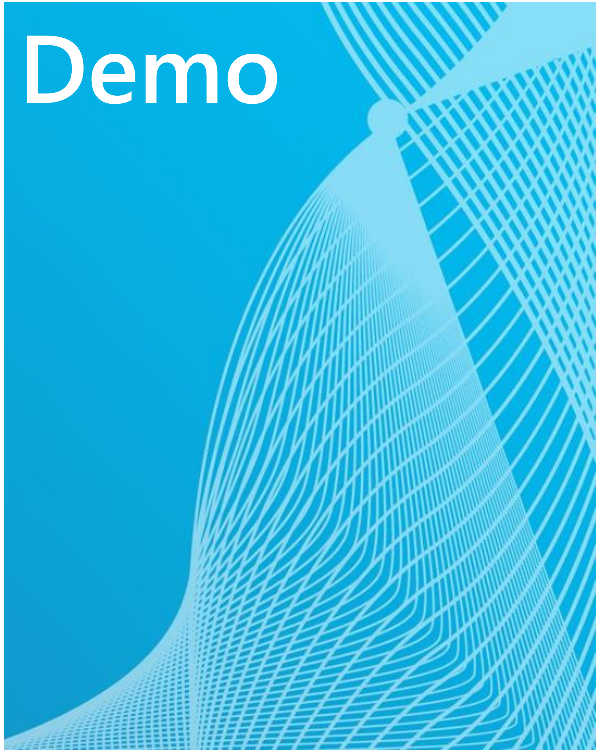
Visual Studio Code

Further readings

- Building middlewares

https://github.com/rstropek/Samples/tree/master/AspNetCoreWorkshop/50-simplest-aspnet

Walkthrough VS “File – New – Project”



Create web project in VS2015

Walkthrough

- Servers (IIS and Kestrel)
- Environments
- Adding MVC

101 for ASP.NET Core

Application Startup

- `Main` Method
- `Startup` class with `ConfigureServices` (DI) and `Configure` (Pipeline)

Static Files

Environments

Servers

- IIS, Kestrel

Configuration

No **`web.config`** anymore

Key/value pair settings from different providers

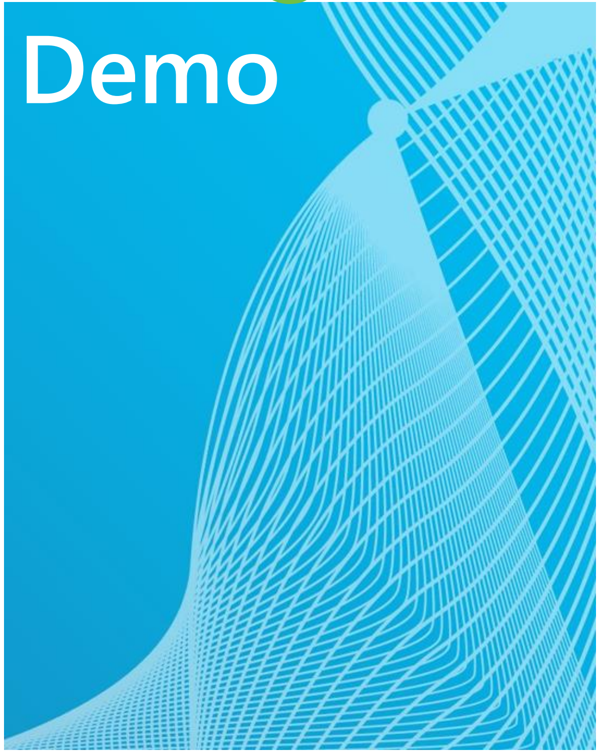
- E.g. memory, environment variables, JSON, INI, XML

Extensible

- Details about writing custom providers

Options pattern for DI integration

Configuration



Demo

In-memory configuration

JSON configuration

Configuration via command line

Configuration with environment variables

Options pattern

See practical use in AppInsights

https://github.com/rstropek/Samples/tree/master/AspNetCoreWorkshop/55-configuration/

Logging

Support for logging built into ASP.NET Core

Various logger built in

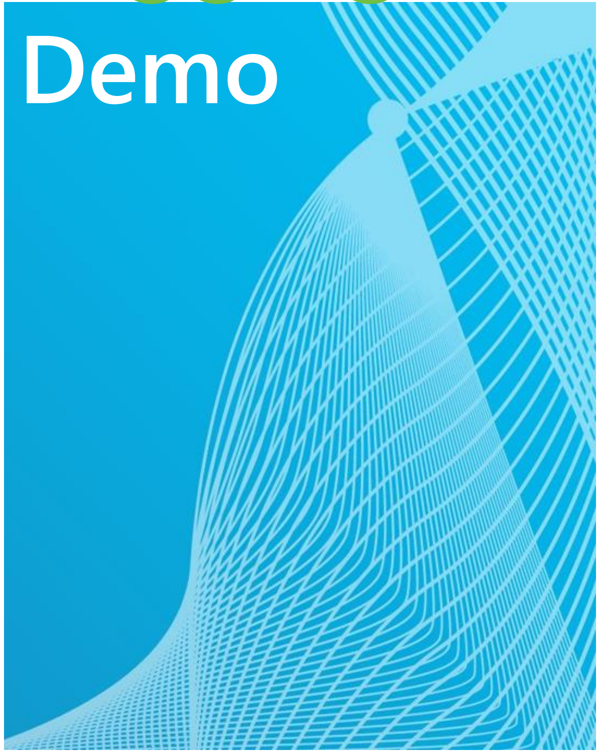
- E.g. console, NLog

Details about logging

Consider using Application Insights

- Getting started with AppInsights in ASP.NET Core

Logging



Demo

JSON file to configure logging

- .NET Core Logging
- AppInsights

Custom logging

AppInsights portal

https://github.com/rstropek/Samples/tree/master/AspNetCoreWorkshop/58-logging/

Dependency Injection

Support for DI built into ASP.NET Core

- Details about DI

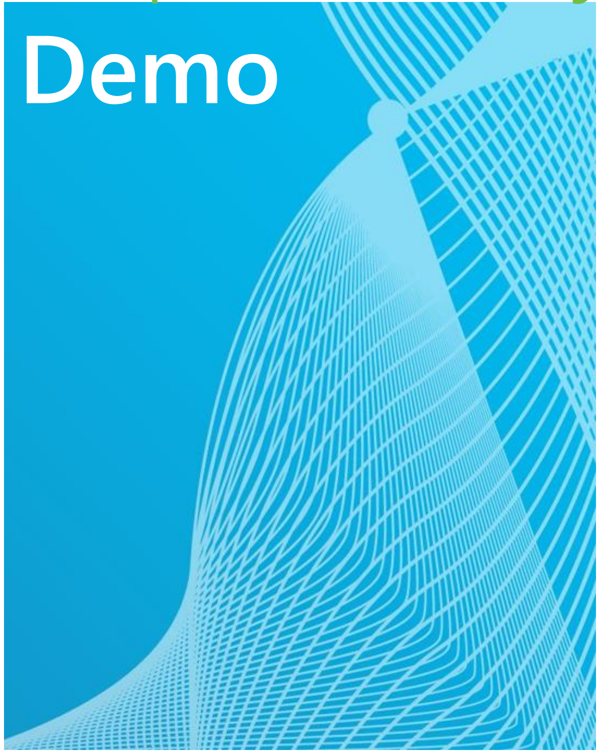
Framework-provided services and your own services

Service Lifetime

- Transient, Scoped, Singleton, Instance

Default container can be replaced (details)

Dependency Injection



Demo

Setting up DI

Service Lifetime

https://github.com/rstropek/Samples/tree/master/AspNetCoreWorkshop/60-di-scopes/

.NET Core Automation

Test, build, and release automation

CI with .NET Core apps

VSTS supports building and publishing .NET Core apps

- Details

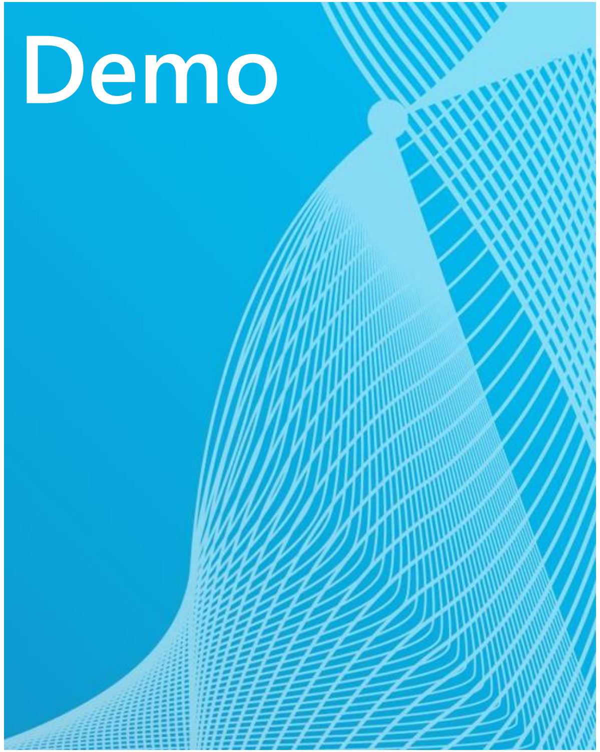
Azure App Services supports .NET Core apps

- Kudu-support for .NET Core

Ready-made Docker image with `Dockerfile`

- `microsoft/dotnet`

Build Automation

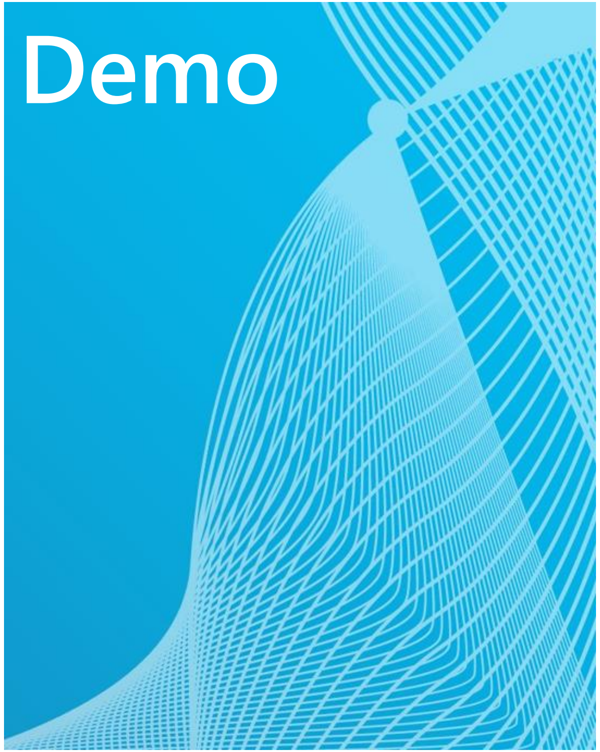


Demo

Build and deploy .NET Core in VSTS

https://www.visualstudio.com/en-us/docs/build/apps/aspnet/aspnetcore-to-azure

Dockerfile for .NET Core app

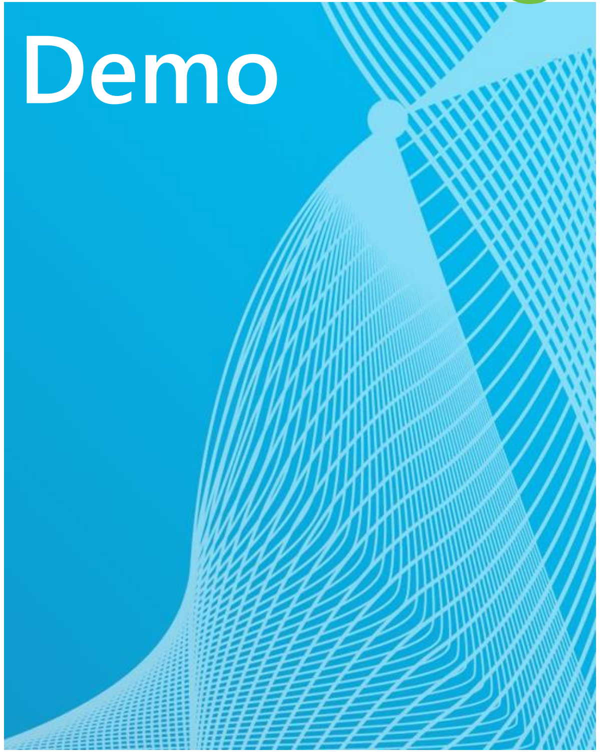


Demo

Unit Testing

- .NET Core supports multiple test frameworks
  - E.g. XUnit, MSTest
  - Compare XUnit and MSTest

Unit Testing



Demo

Create and run library with tests

- XUnit (sample)
- MSTest (sample)

Run tests with

```
dotnet xunit
dotnet test
```

See also https://xunit.github.io/docs/getting-started-dotnet-core

C# 7

Live Coding; sample code see https://github.com/rstropek/Samples/tree/master/CSharp7

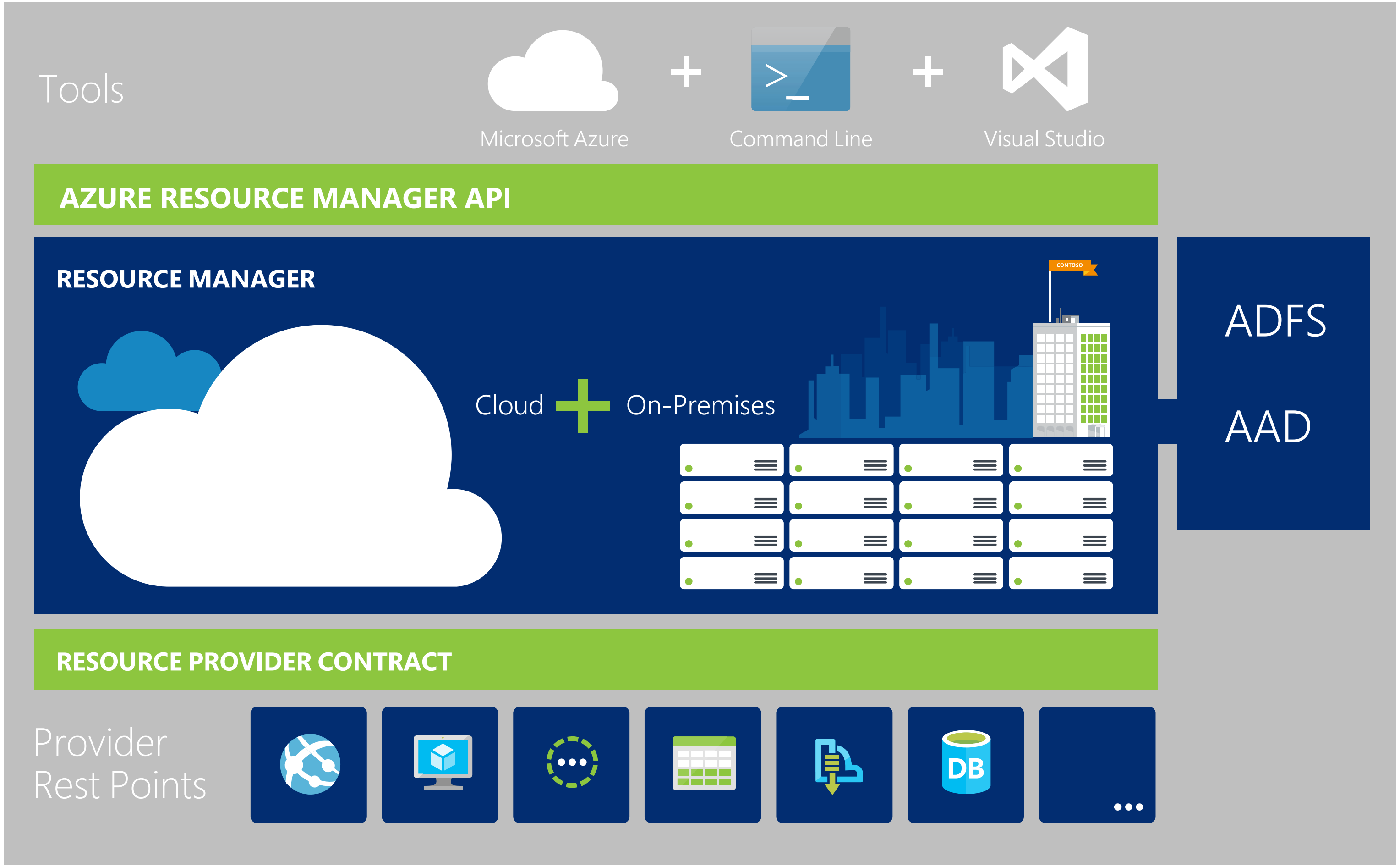
Azure Resource Manager

Why do we need ARM?

In the Early Days…

- *Azure Service Management API* was the version 1 that provided programmatic access for functionality in the Azure platform
- Very limited functionality
  - Examples: ASM can be used to configure Cloud Services, Storage accounts, Virtual Networks
  - No way to target multi-region or multi-service in a single script
- No consistency in the API exposed by services
  - XML, some used JSON
- Limited access control
  - Subscription co-administrator for providing user access
- Limited auditing available from the portal
- Hard to organize lots of resources across the organization

Consistent Management Layer



Tools

Microsoft Azure + Command Line + Visual Studio

AZURE RESOURCE MANAGER API

RESOURCE MANAGER

Cloud + On-Premises

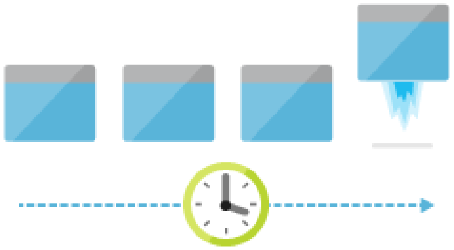
ADFS

AAD

RESOURCE PROVIDER CONTRACT

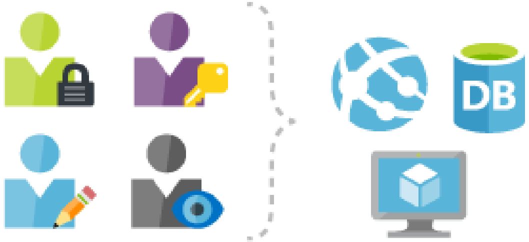
Provider Rest Points

Areas of Focus



Deploy

Organize



Control

Deploying with ARM

template-driven

declarative

idempotent

multi-service

multi-region

extensible

Support for IaaS and PaaS

Support for IaaS

- Incl. Networking

Support for PaaS

Mixed environments

- E.g. web app in IaaS, SQL DB in PaaS

Resources

Resource Manager Overview

https://azure.microsoft.com/en-us/documentation/articles/resource-group-overview/

Supported Services

https://azure.microsoft.com/en-us/documentation/articles/resource-manager-supported-services/

Template Language Reference

https://azure.microsoft.com/en-us/documentation/articles/resource-group-authoring-templates/

Advanced Concepts, Resources

Template functions

E.g. string functions, numeric functions, array functions, deployment values, etc.

https://azure.microsoft.com/en-us/documentation/articles/resource-group-template-functions/

Template linking

https://azure.microsoft.com/en-us/documentation/articles/resource-group-linked-templates/

Creating multiple instances

https://azure.microsoft.com/en-us/documentation/articles/resource-group-create-multiple/

Best Practices

https://azure.microsoft.com/en-us/documentation/articles/best-practices-resource-manager-design-templates/

Summary

Infrastructure is code

ARM makes Azure ready for large-scale

- Number of resources, regions, etc.

ARM makes management easier

- E.g. idempotency, tags, access control

ARM is cross-platform

- PowerShell, Azure CLI, or REST
- Create Linux and Windows resources

Serverless

An Introduction

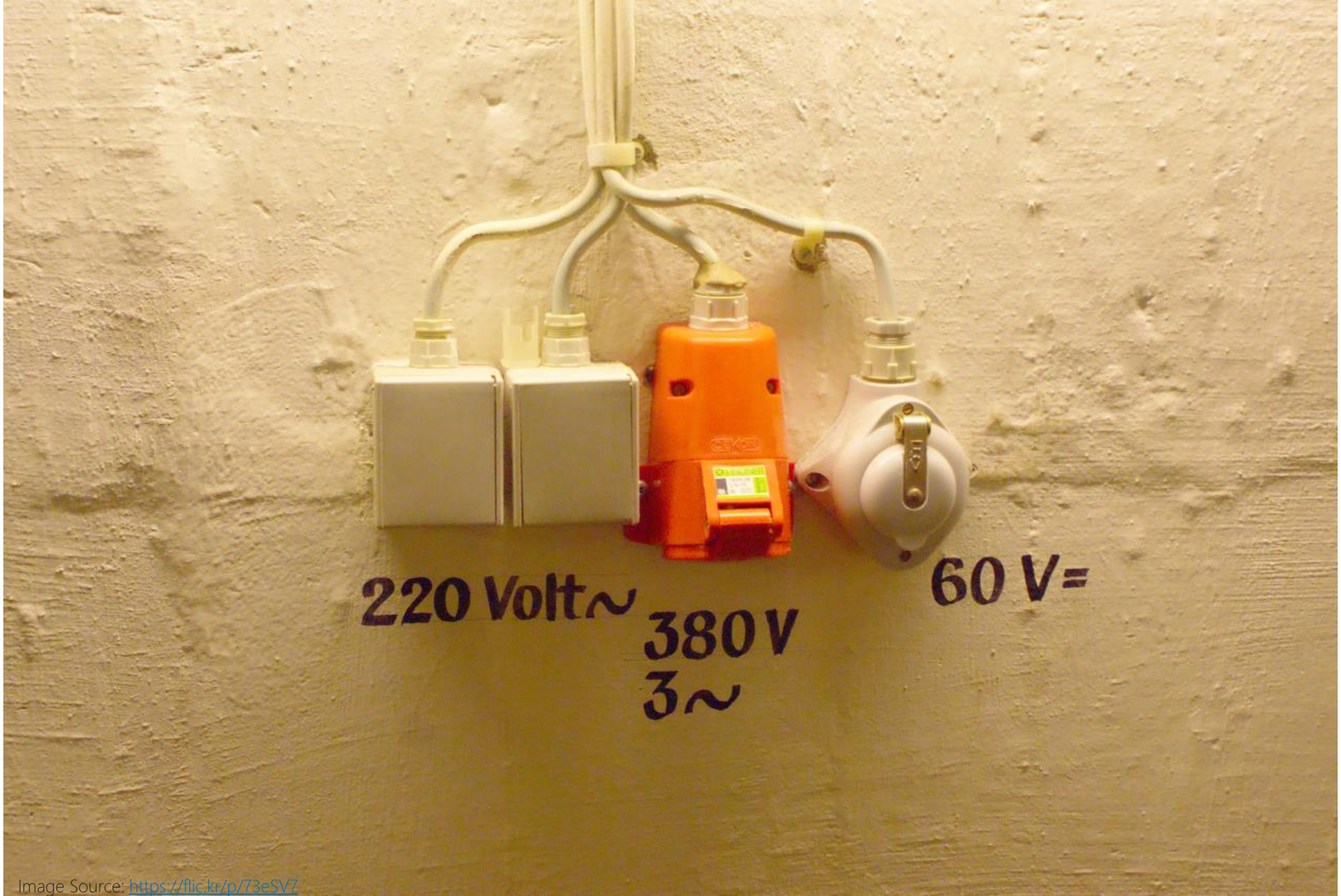
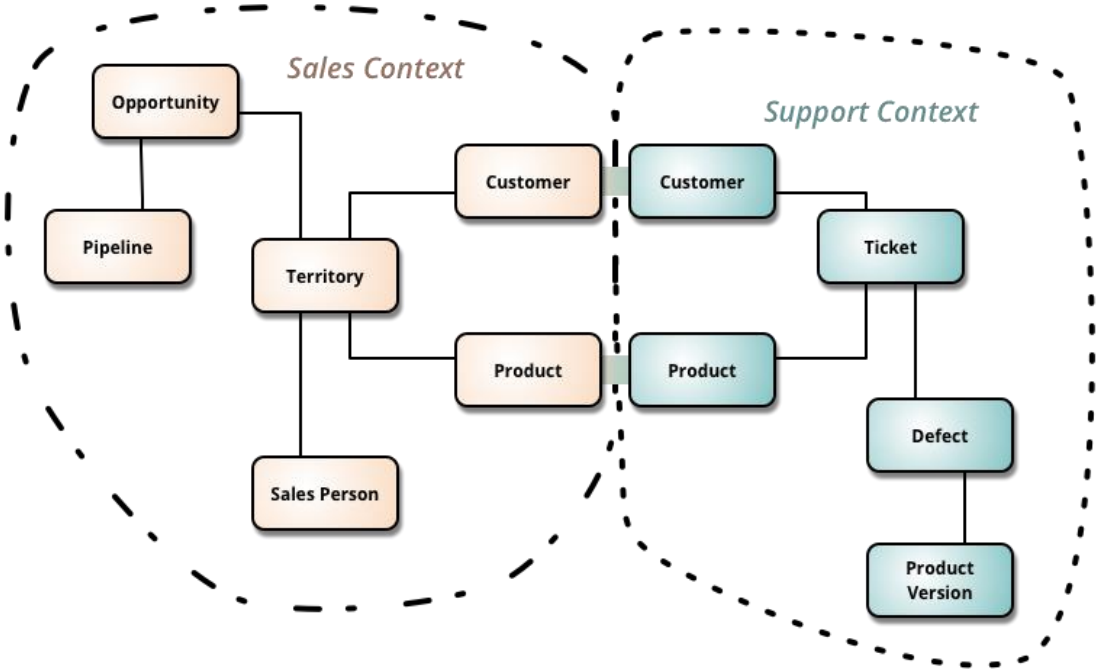


Image Source: https://flic.kr/p/73eSV7

Microservices



Source: http://martinfowler.com/bliki/BoundedContext.html

Autonomous Teams

Image Source: https://flic.kr/p/4JGGFW

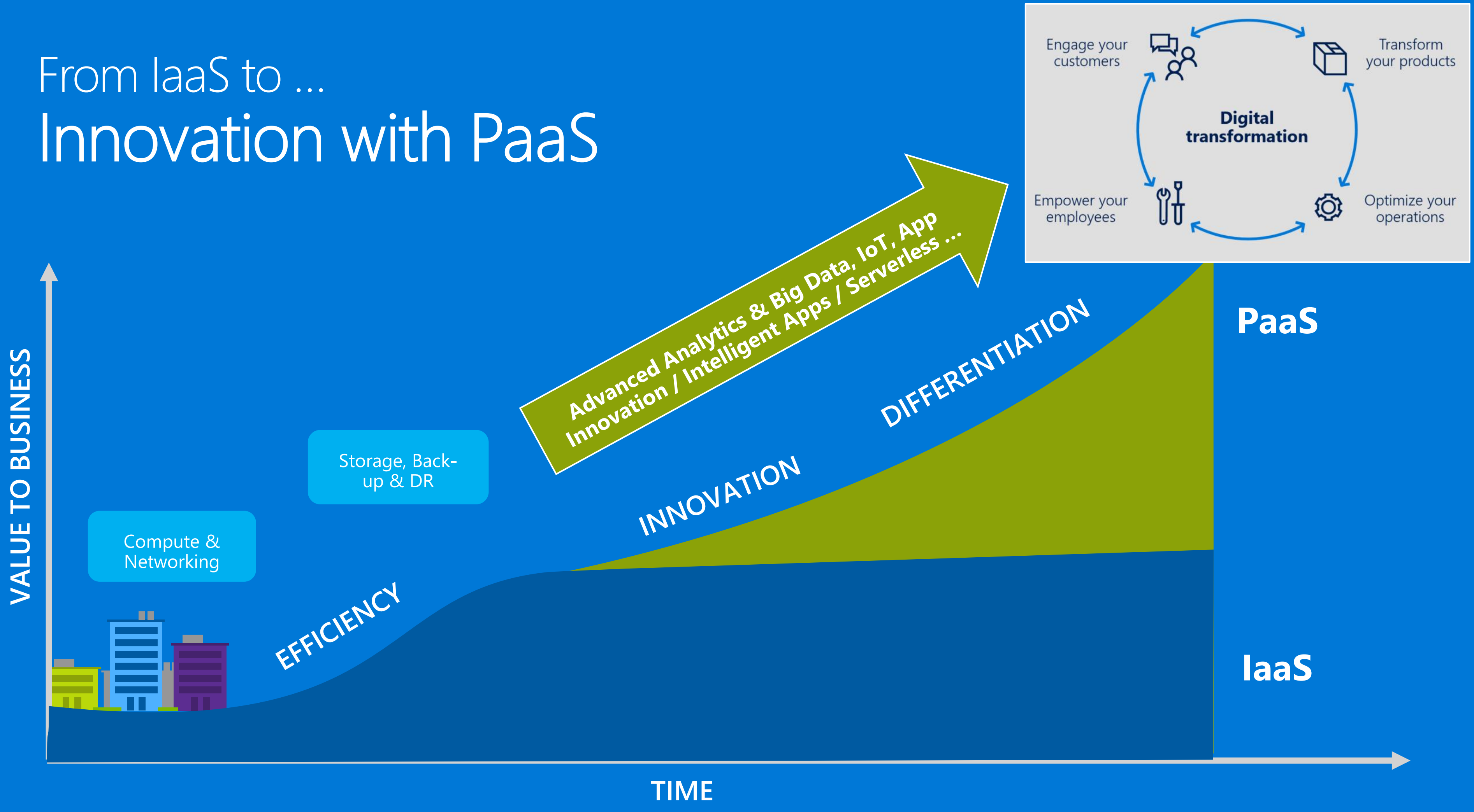
Image Source: https://flic.kr/p/5VB88j

Image Source: https://flic.kr/p/65RcDT

Image Source: https://flic.kr/p/5qtj4r

CoderDojo

From IaaS to …
Innovation with PaaS



Engage your customers

Transform your products

Digital transformation

Empower your employees

Optimize your operations

Advanced Analytics & Big Data, IoT, App Innovation / Intelligent Apps / Serverless …

VALUE TO BUSINESS

Storage, Back-up & DR

Compute & Networking

EFFICIENCY

INNOVATION

DIFFERENTIATION

PaaS

IaaS

TIME

Why Not?

Trust is Required

Image Source: https://flic.kr/p/9nWs1

Specific Needs

Image Source: https://flic.kr/p/gWCWqU

Azure Functions

Getting Started

Functions ≈ WebJobs on steroids

- Scripting, Web UI

Functions are implemented using Azure App Services

- Good to be familiar with App Services when working with Functions

Azure Functions Characteristics

Choice of language

- C#, F#, TypeScript, etc.

Pay-per-use pricing model

- Dynamic App Service Plan

Support for NuGet and NPM

Integrated security

- Support for OAuth providers like AAD, Facebook, Google, Twitter, and Microsoft Account

Code-less integration

Flexible development

- In-portal editor or set up continuous integration (e.g. GitHub, VSTS, local Git repository)

Triggers & Bindings

Timer

HTTP (Web Host)

- REST, Webhook

Azure Storage

- Blobs, Queues, Tables

Service Bus

- Queues, Topics

DocumentDB

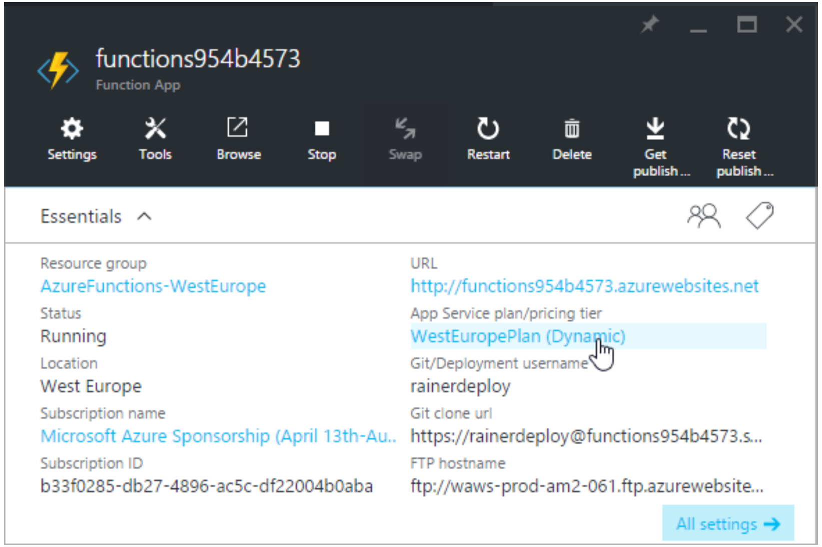
Details

Dynamic App Service Plan

Only pay for the time that your code spends running

Functions pricing (based on “GB-s”, “Gigabyte Seconds”)

„nearest 100ms at Per/GB price based on the time your function runs and the memory size of the function space you choose”



Resource group
AzureFunctions-WestEurope

Status
Running

Location
West Europe

Subscription name
Microsoft Azure Sponsorship (April 13th-Au..

Subscription ID
b33f0285-db27-4896-ac5c-df22004b0aba

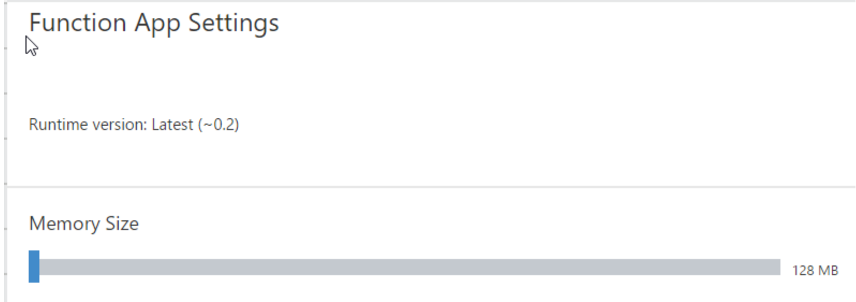
URL
http://functions954b4573.azurewebsites.net

App Service plan/pricing tier
WestEuropePlan (Dynamic)

Git/Deployment username
rainerdeploy

Git clone url
https://rainerdeploy@functions954b4573.s...

FTP hostname
ftp://waws-prod-am2-061.ftp.azurewebsite...



Function App Settings

Runtime version: Latest (~0.2)

Memory Size
128 MB

C# Function

TicTacToe Logic

Create *Function App*

- Consumption Plan
- Application Insights integration

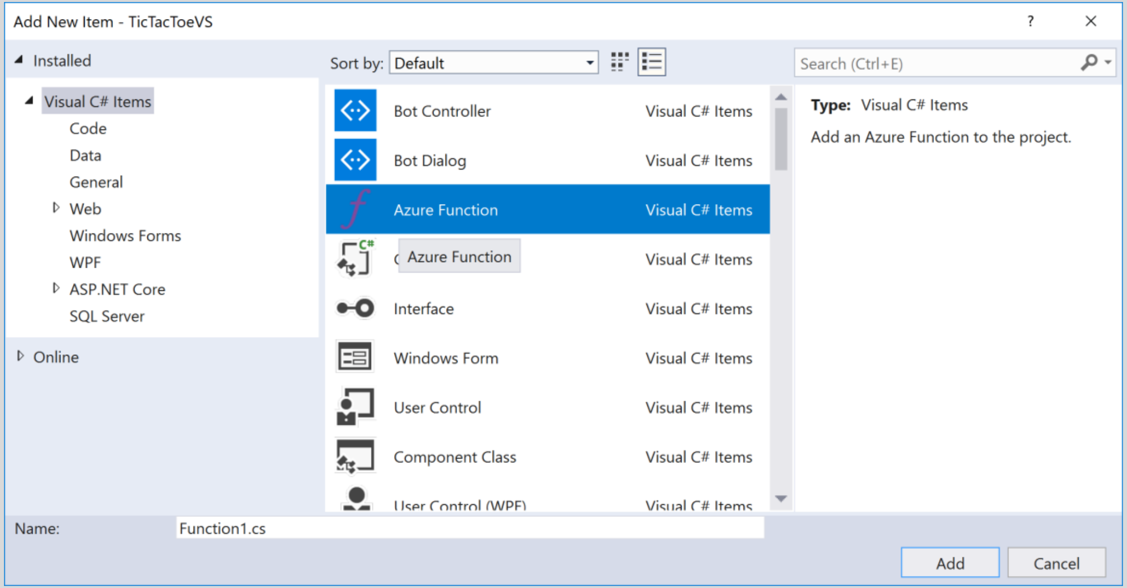
Create in Portal

Call via HTTP (Insomnia)

Show background files

- C# Scripts
- JSON configuration files

```
using System.Net;

public static async Task<HttpResponseMessage> Run(HttpRequestMessage req, TraceWriter log) {
    log.Info("Received Tic-Tac-Toe request");
    var board = await req.Content.ReadAsAsync<string[]>();

    if (board.Length != 9) {
        return req.CreateResponse(HttpStatusCode.BadRequest, "No valid tic-tac-toe board");
    }

    for(var row = 0; row < 3; row ++) {
        if (!string.IsNullOrWhiteSpace(board[row * 3])
            && board[row * 3] == board[row * 3 + 1] && board[row * 3] == board[row * 3 + 2]) {
                return BuildResponse(req, board[row * 3]);
            }
    }

    for(var column = 0; column < 3; column ++) {
        if (!string.IsNullOrWhiteSpace(board[column])
            && board[column] == board[3 + column] && board[column] == board[2 * 3 + column]) {
                return BuildResponse(req, board[column]);
            }
    }

    if (!string.IsNullOrWhiteSpace(board[0])
        && board[0] == board[3 + 1] && board[0] == board[2 * 3 + 2]) {
            return BuildResponse(req, board[0]);
        }

    if (!string.IsNullOrWhiteSpace(board[2])
        && board[2] == board[3 + 1] && board[2] == board[2 * 3]) {
            return BuildResponse(req, board[1]);
        }

    return BuildResponse(req);
}

private static HttpResponseMessage BuildResponse(HttpRequestMessage req, string winner = null)
    => req.CreateResponse(HttpStatusCode.OK, new { winner = winner });
```



Type: Visual C# Items

Add an Azure Function to the project.

Name: Function1.cs

C# Function

Develop in Visual Studio

Create *Function App*

Visual Studio

Debugging with VS

Deploy using VS

```
func init --no-source-control
func templates list
func new
  -> Answer wizard questions (Node.js)

func start
```

Node.js Function

Local Development

Create Function App

Node.js

Call via HTTP (Insomnia)

Node Function

TicTacToe Validator

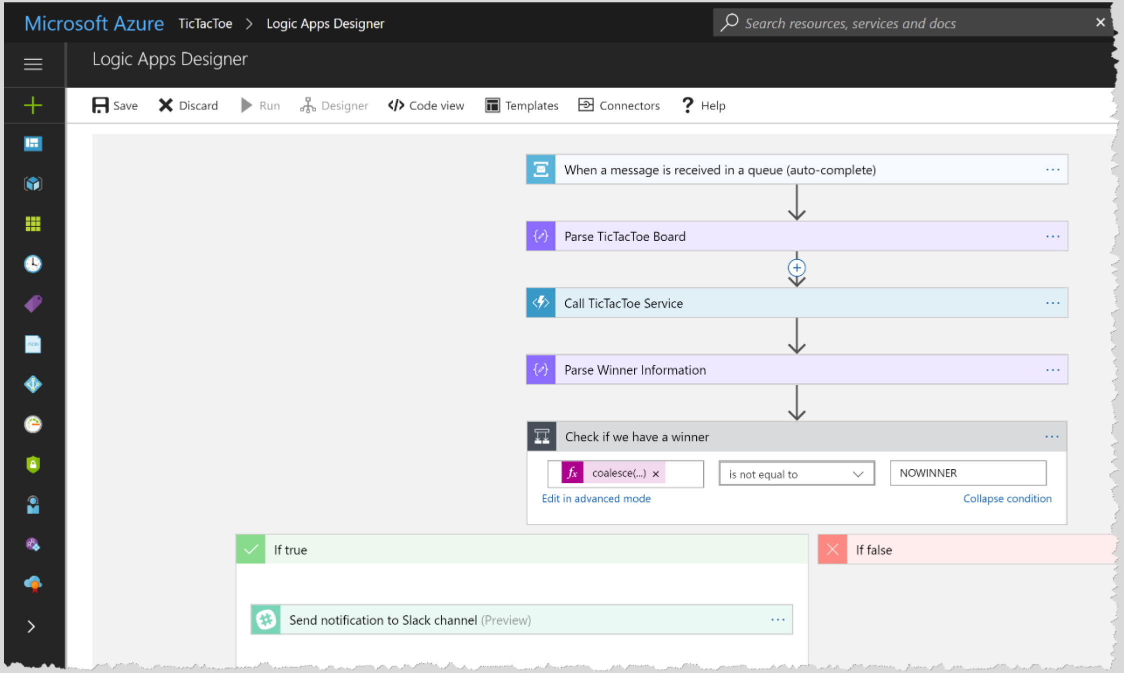
```
module.exports = function (context, req) {
  // Parse request body
  var board = JSON.parse(req.body);

  // Make sure that body is a properly formed array
  if (Array.isArray(board) && board.length == 9) {
      // Body is ok -> send message to trigger analysis
      context.bindings.outputSbMsg = { Message: board };

      // Send OK result to caller
      context.res = { status: 202 };
      context.done();
  }
  else {
      // Body is malformed -> send Bad Request to caller
      context.res = { status: 400, body: "No valid tic-tac-toe board" };
      context.done();
  }
};
```

Create Queue Binding

Call via HTTP (Insomnia)

Logic App

Combine Functions with Workflows



When a message is received in a queue (auto-complete)

Parse TicTacToe Board

Call TicTacToe Service

Parse Winner Information

Check if we have a winner

coalesce(...) | is not equal to | NOWINNER

Edit in advanced mode

Collapse condition

If true

Send notification to Slack channel (Preview)

If false

Create function

Connect with Service Bus

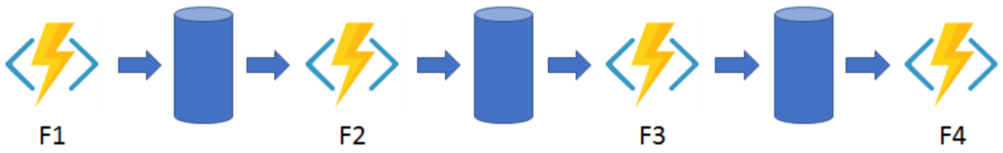
Views

Design View

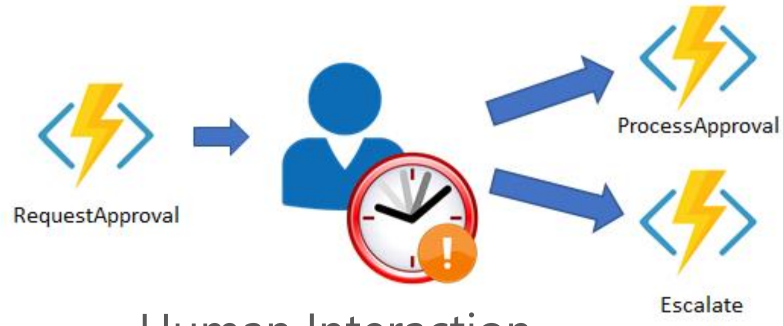
Code View

What Else?

Durable Functions (Preview)



Function Chaining



Human Interaction

https://docs.microsoft.com/en-us/azure/azure-functions/durable-functions-overview

Summary

Summary

It's all about Microservices

Specialized services working together to form a customer solution

Resources on demand

How much does your software cost if it is in standby?

Idempotency

There are no transactions → when in doubt, send again

Prepare for failures

Out-of-sync issues, unreliable networks, servers constantly change, etc.

Importance of logging and telemetry

Learn and implement OpenID Connect

AAD is an easy-to-use option

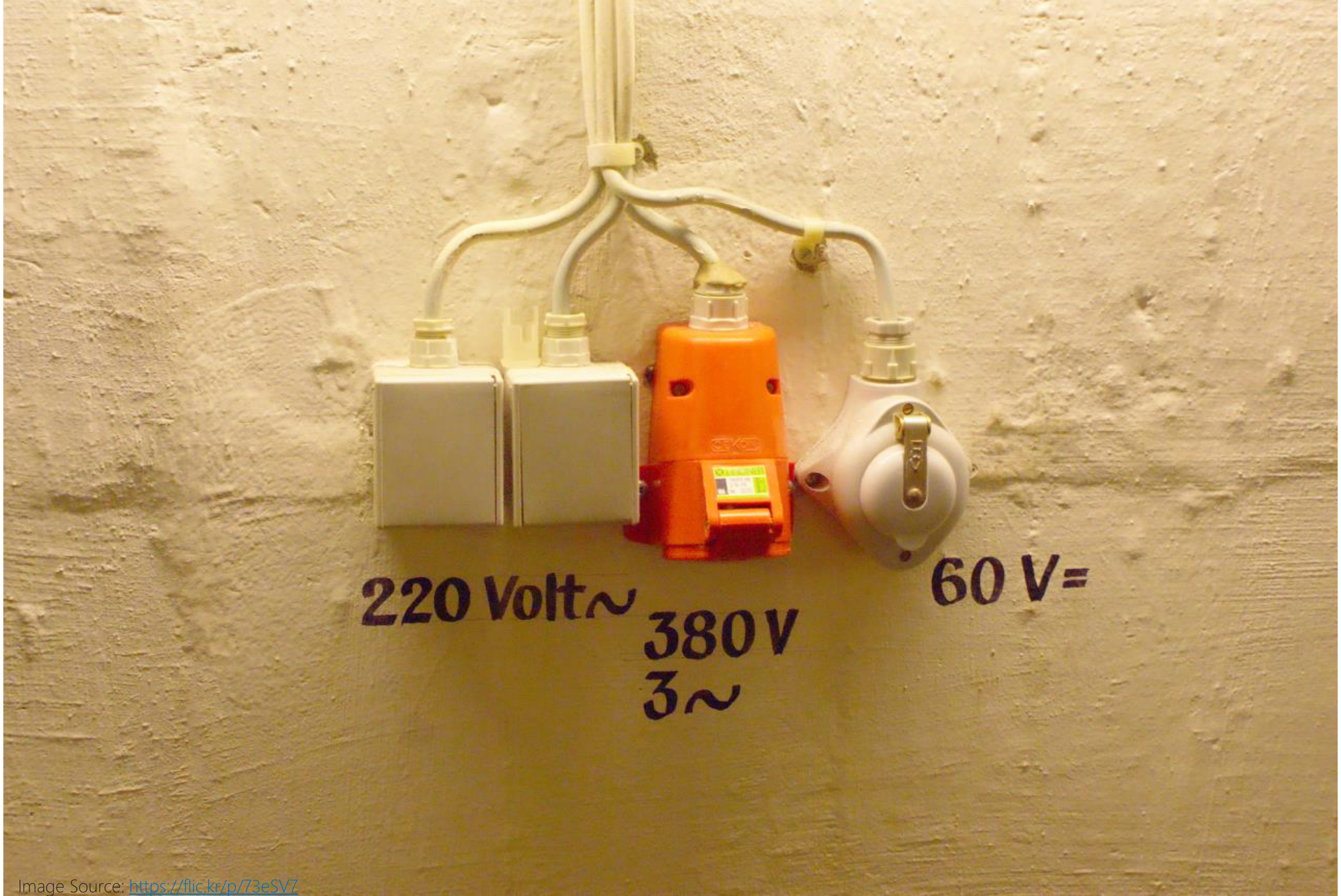


Image Source: https://flic.kr/p/73eSV7

.NET and Docker

An Introduction

Everybody knows Docker?

Do we need a whirlwind recap?

Docker Images

.NET Core

- https://hub.docker.com/r/microsoft/dotnet/

ASP.NET Core

- https://hub.docker.com/r/microsoft/aspnetcore/
- https://hub.docker.com/r/microsoft/aspnetcore-build/

.NET 3.5 and 4.x

- https://hub.docker.com/r/microsoft/dotnet-framework/

ASP.NET 3.x and 4.x

- https://hub.docker.com/r/microsoft/aspnet/

CI/CD

- https://hub.docker.com/r/microsoft/vsts-agent/

Read more…

Samples from this talk

https://github.com/rstropek/DockerVS2015Intro/tree/master/dockerDemos/08-docker-dot-net

ASP.NET and Docker

https://docs.microsoft.com/en-us/aspnet/mvc/overview/deployment/docker-aspnetmvc

Dockerize .NET Core

https://docs.docker.com/engine/examples/dotnetcore/

Time cockpit eLearning Library

Q&A

Thank your for coming!

Rainer Stropek

software architects gmbh

Mail rainer@timecockpit.com
Web http://www.timecockpit.com
Twitter @rstropek

time cockpit
Saves the day.