# Developing microservices with .NET Core and Azure DevOps

**Matteo Tumiati**

Windows Development MVP

matteot@aspitalia.com | @xtumiox


aspitalia.com

#netconfit

# Agenda

The developers’ journey

Updates on .NET Core 2.1 “wave” related to services

Microservices patterns

Azure and DevOps

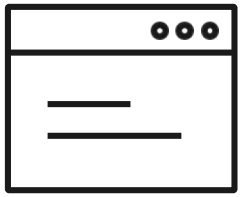
# Going digital

**1 million/hour**
new devices coming online by 2020

**12 years**
average age of S&P 500 corporations by 2020

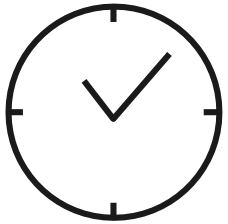
**60%**
computing in the public cloud by 2025

# From the developers…



I need to create applications at a competitive rate **without worrying about IT**

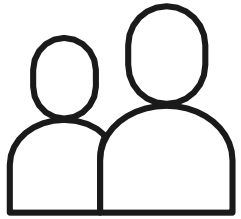
New applications run smoothly on my machine but **malfunction on traditional IT servers**



My productivity and application innovation become suspended when **I have to wait on IT**

# From the IT Pros…

I **need to manage servers** and maintain compliance with little disruption

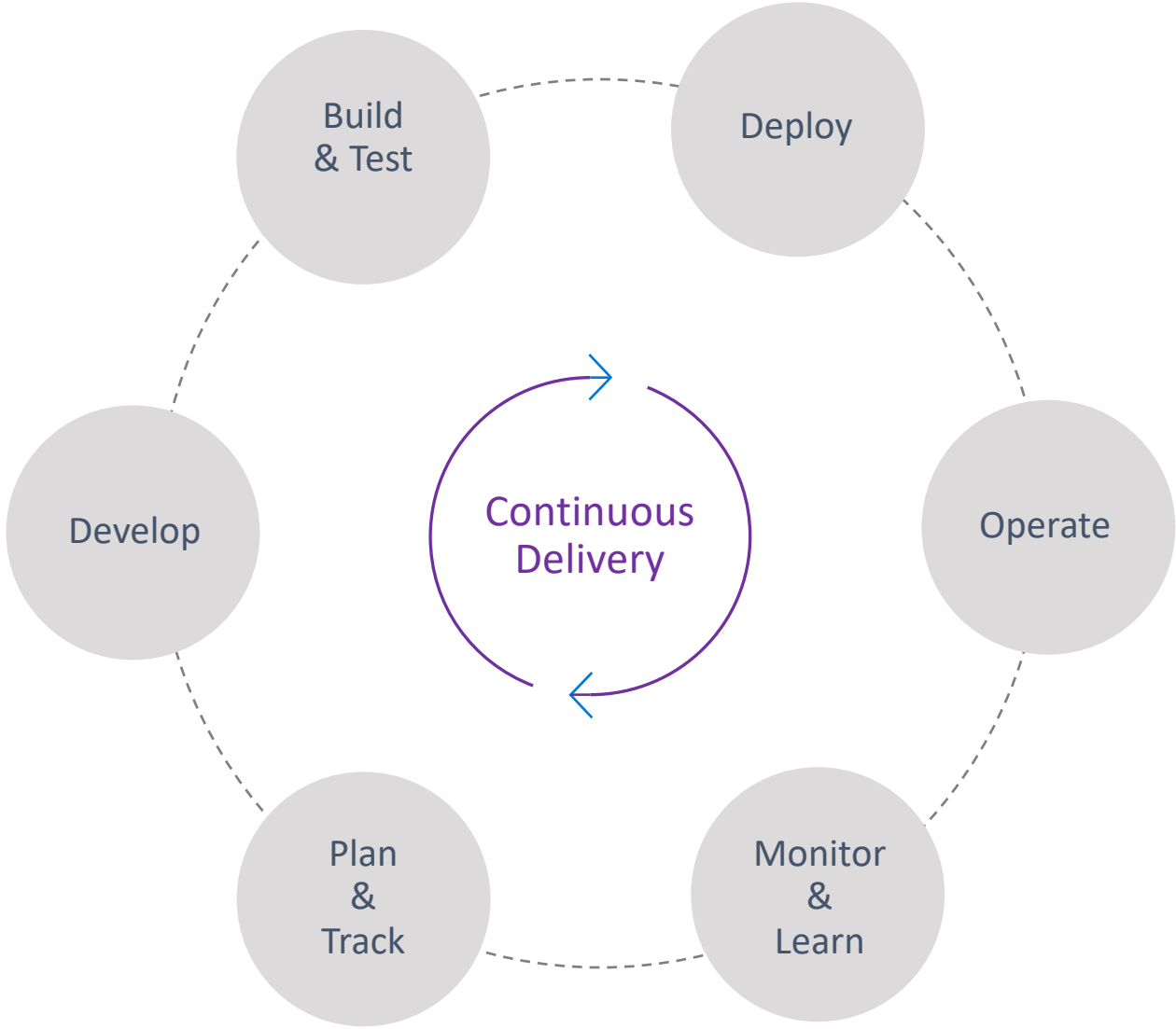


I'm **unsure of how to integrate unfamiliar applications**, and I require help from developers

I'm unable to focus on both **server protection and application compliance**

# What is DevOps?

> DevOps is the union of **people**, **process**, and **products** to enable continuous delivery of value to your end users.



# List of (some) DevOps practices

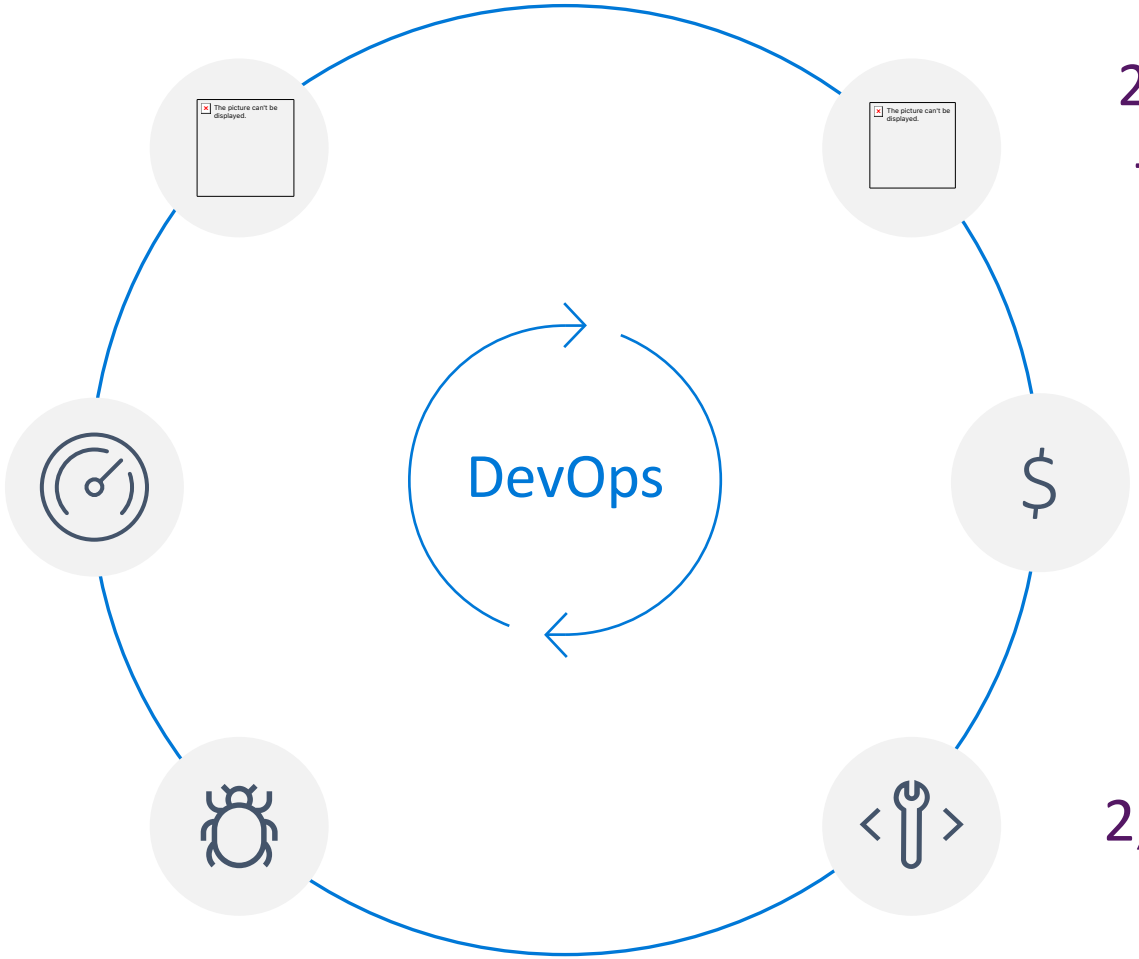
- Infrastructure as Code (IaC)
- Continuous Integration
- Automated Testing
- Continuous Deployment
- Release Management
- App Performance Monitoring
- Load Testing & Auto-Scale
- Availability Monitoring
- Change/Configuration Management
- Automated Environment De-Provisioning
- Self Service Environments
- Automated Recovery
- Feature Toggles
- Hypothesis Driven Development

# High perf companies

46x Deployment Frequency

Faster Time to Market

7x Lower Change Failure Rate




2,555x Faster Lead Time For Changes

Increased Revenue

2,604x Faster Mean Time to Recover

# DevOps at Microsoft

372k
Pull Requests per month

4.4m
Builds per month

5m
Work items viewed per day

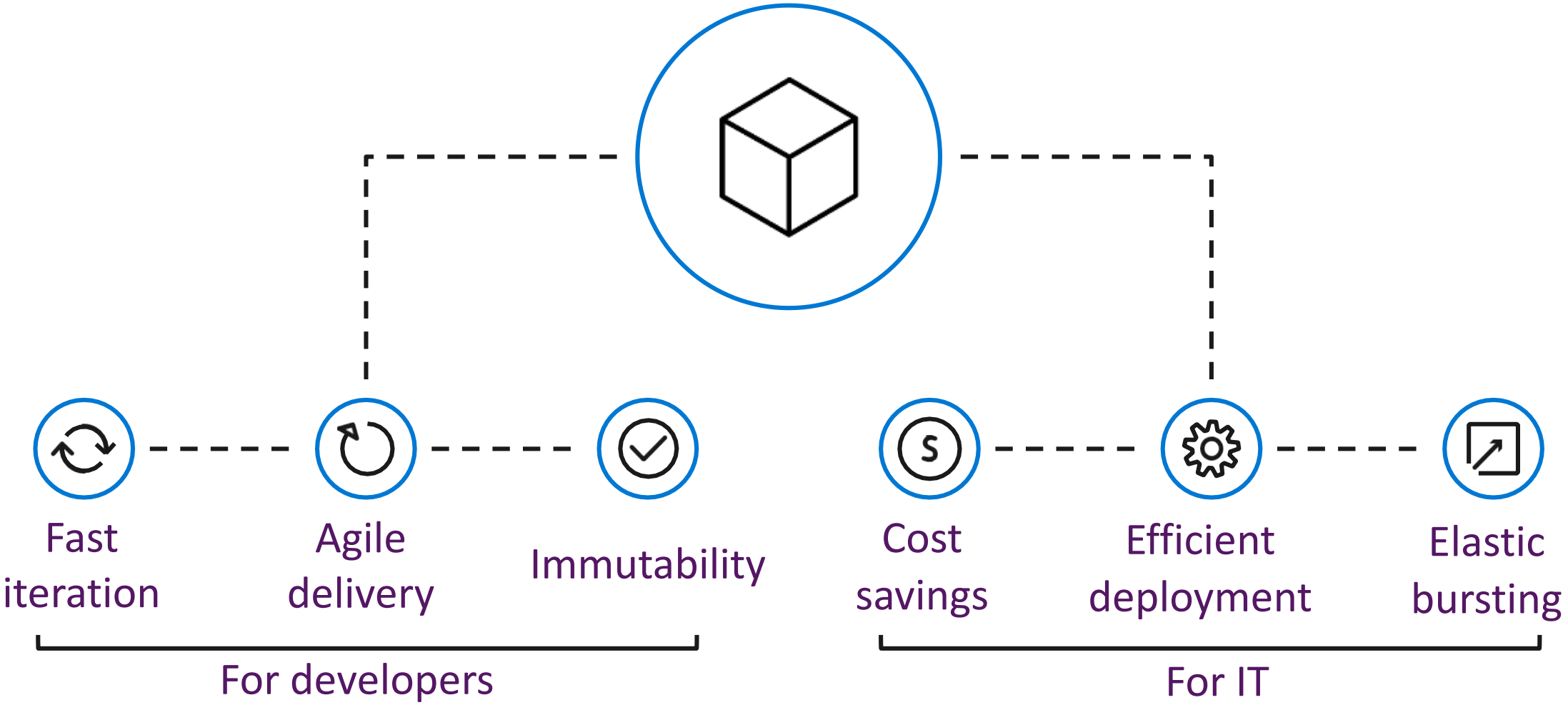
78,000
Deployments per day

2m
Git commits per month

500m
Test executions per day

500k
Work items updated per day

# The container’s approach



# Biggest question ever asked

What is the meaning of life?

Who are we?

Could we be living in a Matrix?

What new particles remain to be discovered?

## What is a container?

# Docker, containers…

## Containers

*Not a real thing.* An application delivery mechanism with process isolation based on several Linux kernel features.

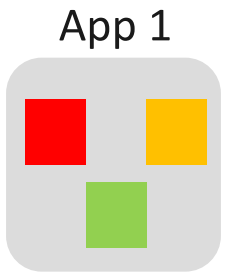
## Docker

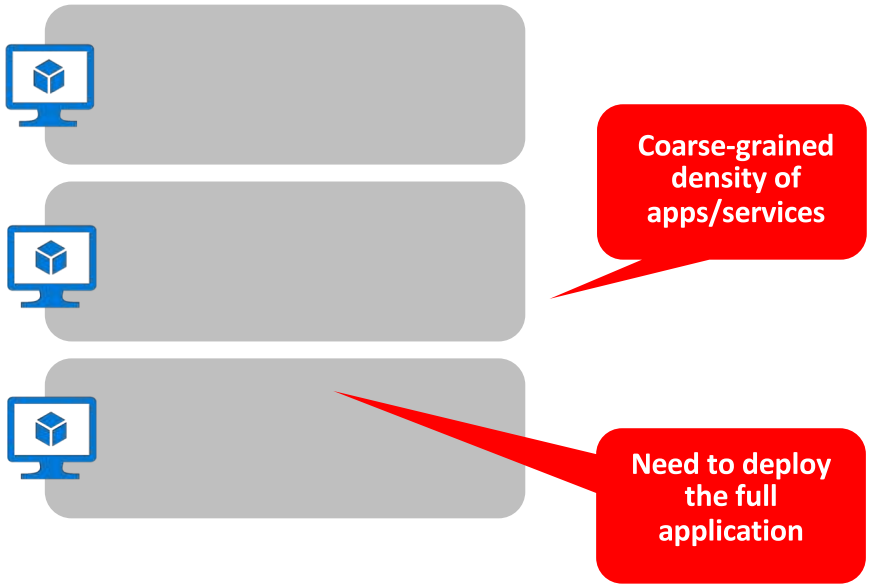
- Open source container runtime
- MacOS, Linux and Windows support
- CLI/Dockerfile

It’s has been a long journey, but it just started…

# Traditional application approach

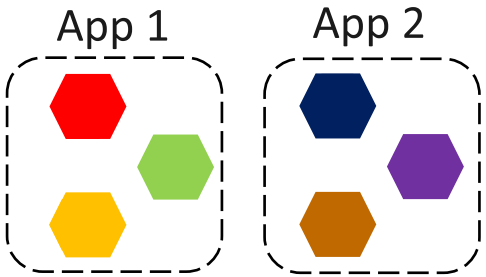
- A traditional application has its functionality within a few processes componentized with layers and libraries
- Scales by cloning the app on multiple servers/VMs

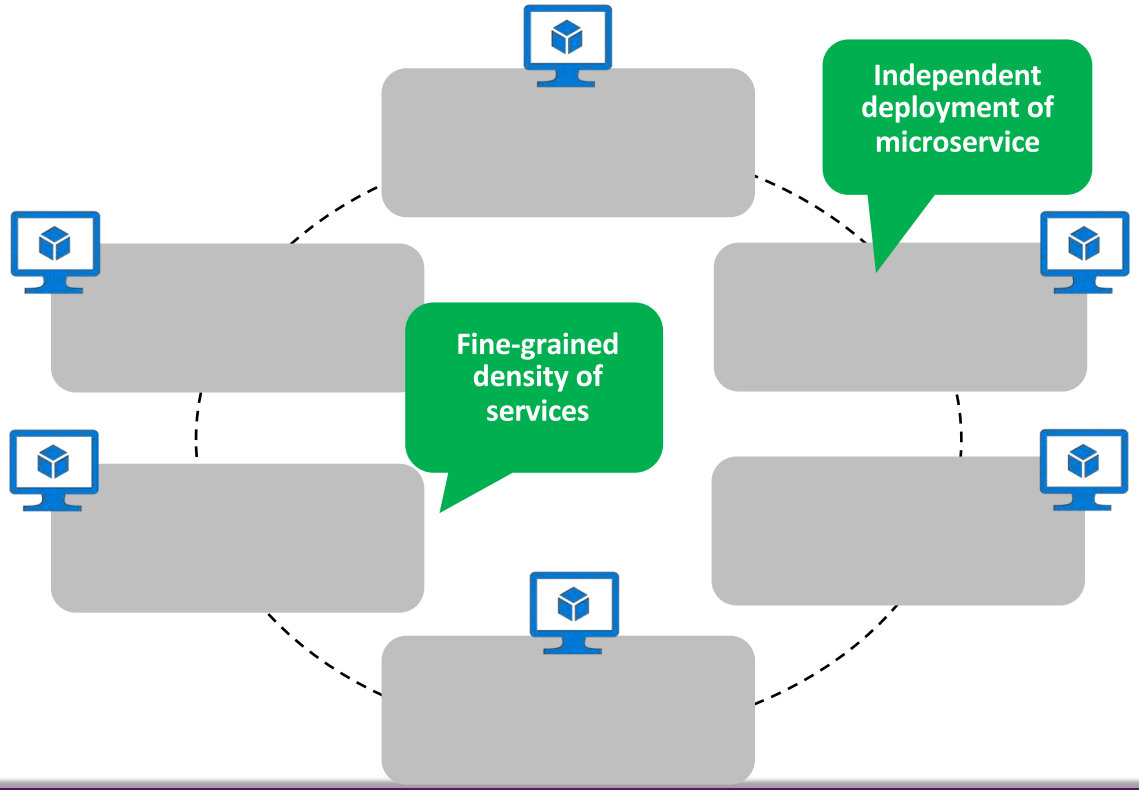



# Microservices application approach

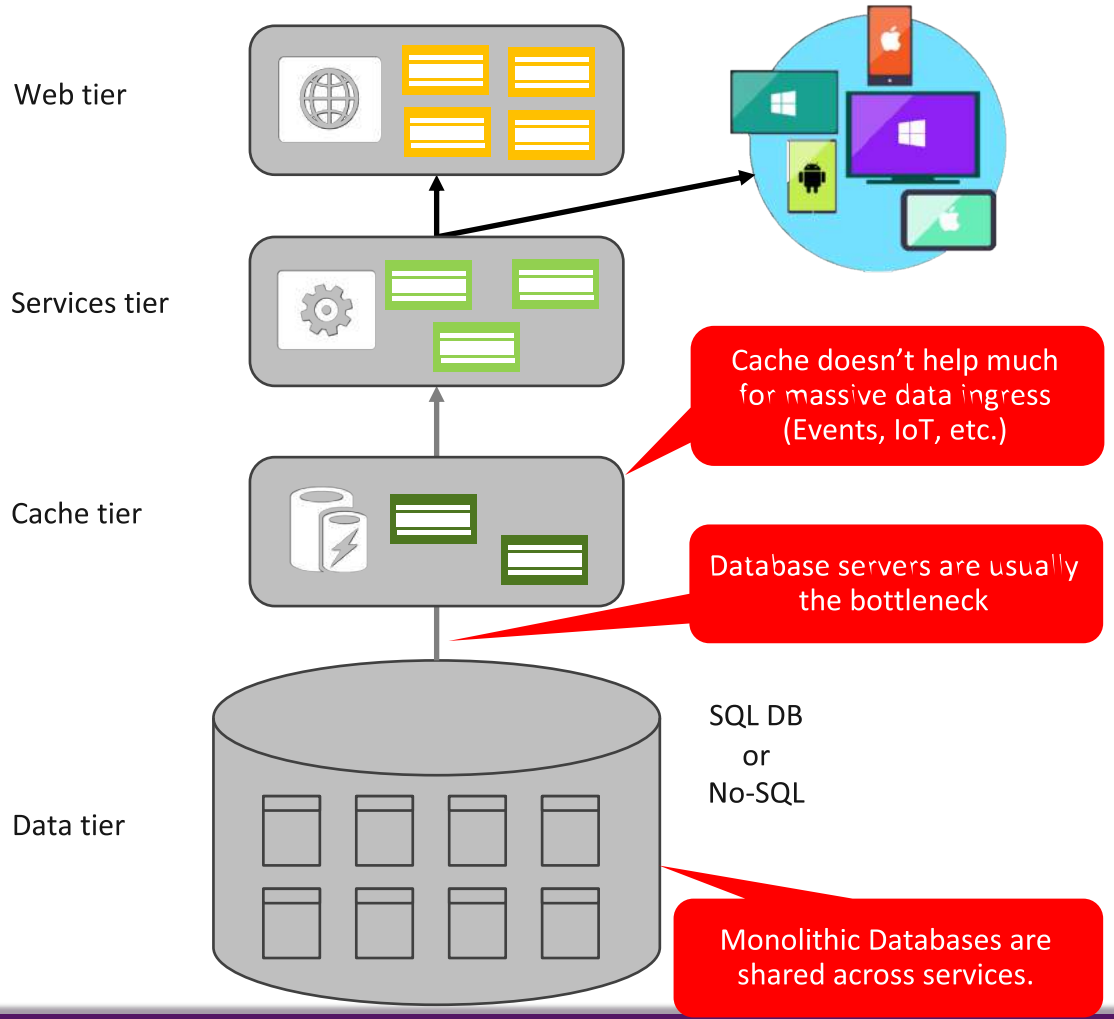
- A microservice application segregates functionality into separate smaller services
- Scales out by deploying each service independently with multiple instances across servers/VMs



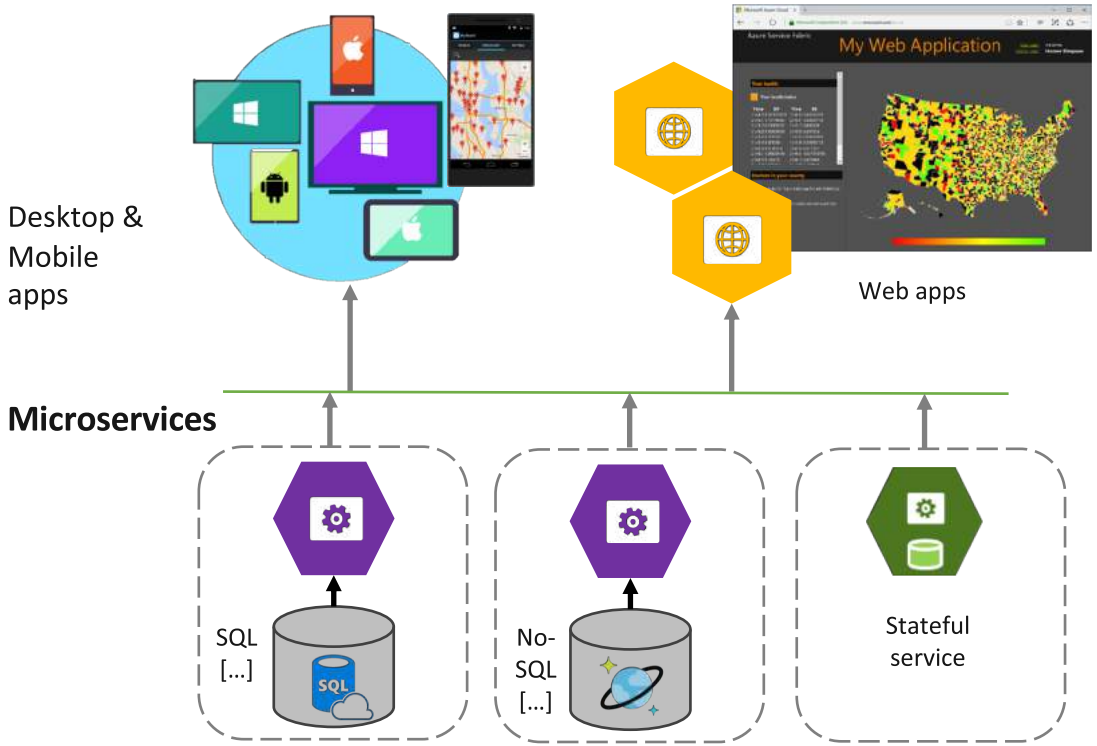
# Traditional application approach

- Single monolithic database
- Tiers of specific technologies



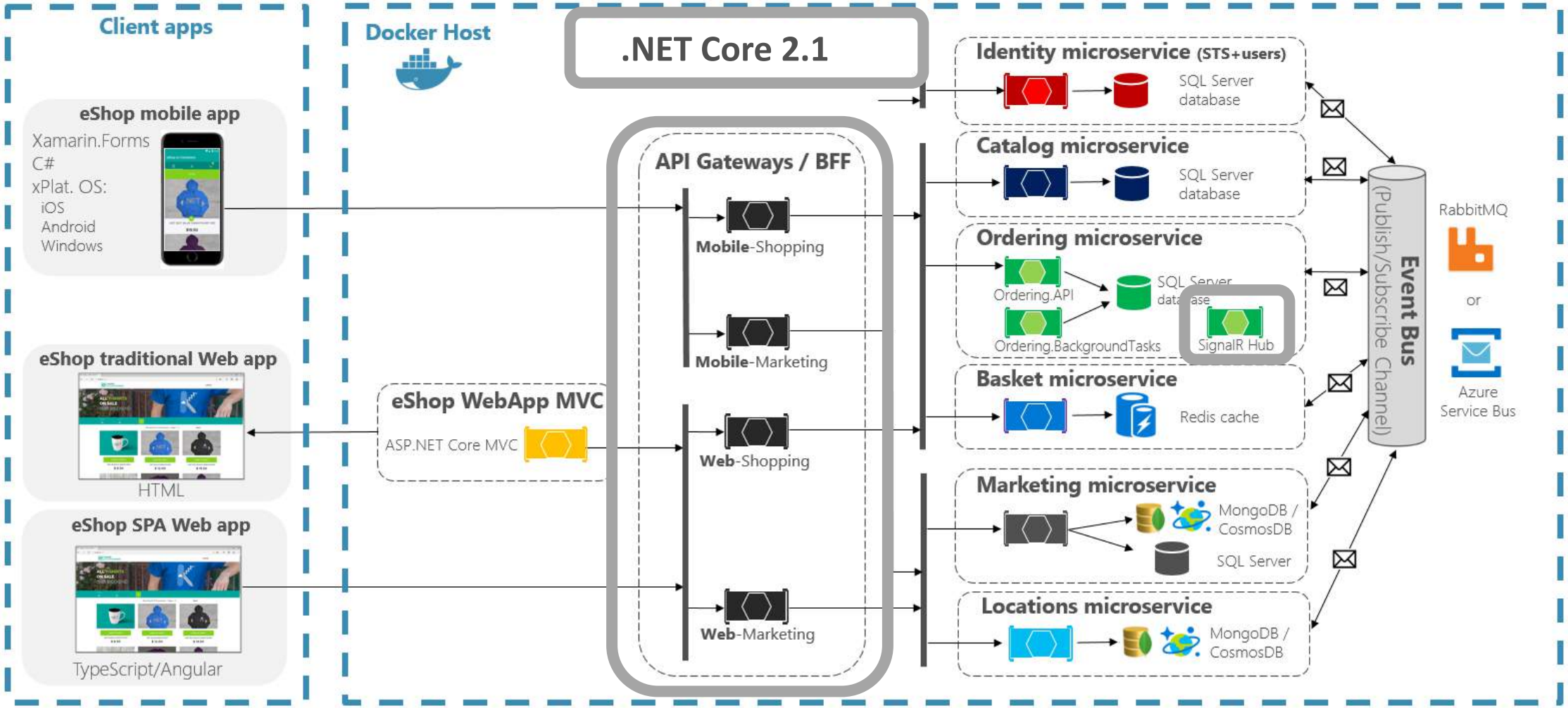

# Data in Microservices approach

- Graph of interconnected microservices
- State/data typically scoped to the microservice
- Remote Storage for cold data



# eShopOnContainers reference application

(Development environment architecture)




https://github.com/dotnet-architecture/eShopOnContainers

# Reference application

Domain Driven Design Patterns

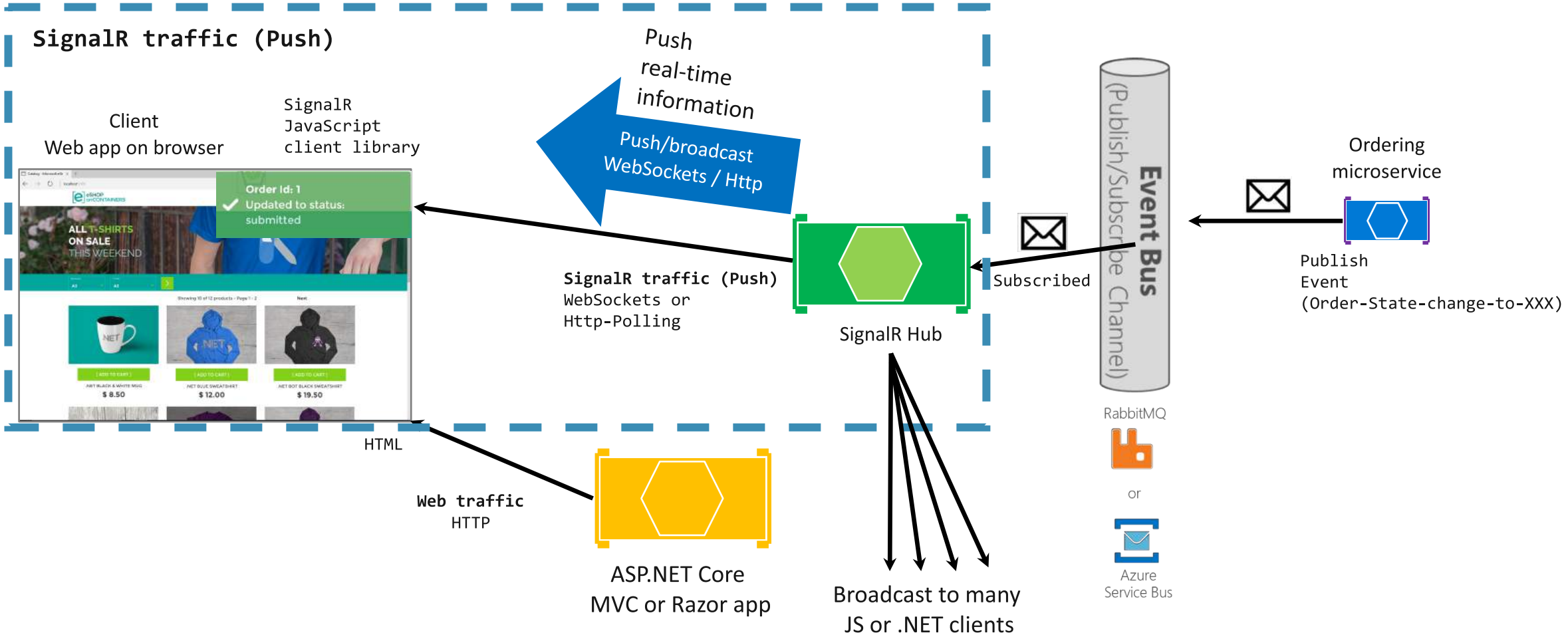
- Domain Models (Aggregates, Entity, Value-Objects, etc.)
- Simplified CQRS
  - Dapper MicroORM for queries
  - Commands and Mediator patter
- Domain Events (within the same microservice)

Microservices

- Integration Events (across microservices)
- Docker Multi-arch and multi-stage builds
- Swagger with Swashbuckle
- Security (AuthN/AuthZ) with tokens from IdentityServer4 wrapping ASP.NET Identity

More info: http://aka.ms/MicroservicesPatterns

# SignalR


SignalR traffic (Push)
Client
Web app on browser
SignalR
JavaScript
client library
Push
real-time
information
Push/broadcast
WebSockets / Http
Order Id: 1
Updated to status:
submitted
ALL T-SHIRTS
ON SALE
THIS WEEKEND
SignalR traffic (Push)
WebSockets or
Http-Polling
SignalR Hub
Subscribed
Event Bus
(Publish/Subscribe Channel)
Ordering
microservice
Publish
Event
(Order-State-change-to-XXX)
RabbitMQ
or
Azure
Service Bus
HTML
Web traffic
HTTP
ASP.NET Core
MVC or Razor app
Broadcast to many
JS or .NET clients

# Demo

End-to-end solution overview in Visual Studio

- Solution’s projects
- Docker-compose.yml
- Helm charts for K8s

Place an order and listen to SignalR notifications

# Common patterns for microservices

Asynchronous pub/subs communication (via Event Bus)

Health checks

Resilient cloud applications:

- Retries with exponential backoff plus circuit breaker with new *HttpClientFactory*

API Gateway vs. Direct communication

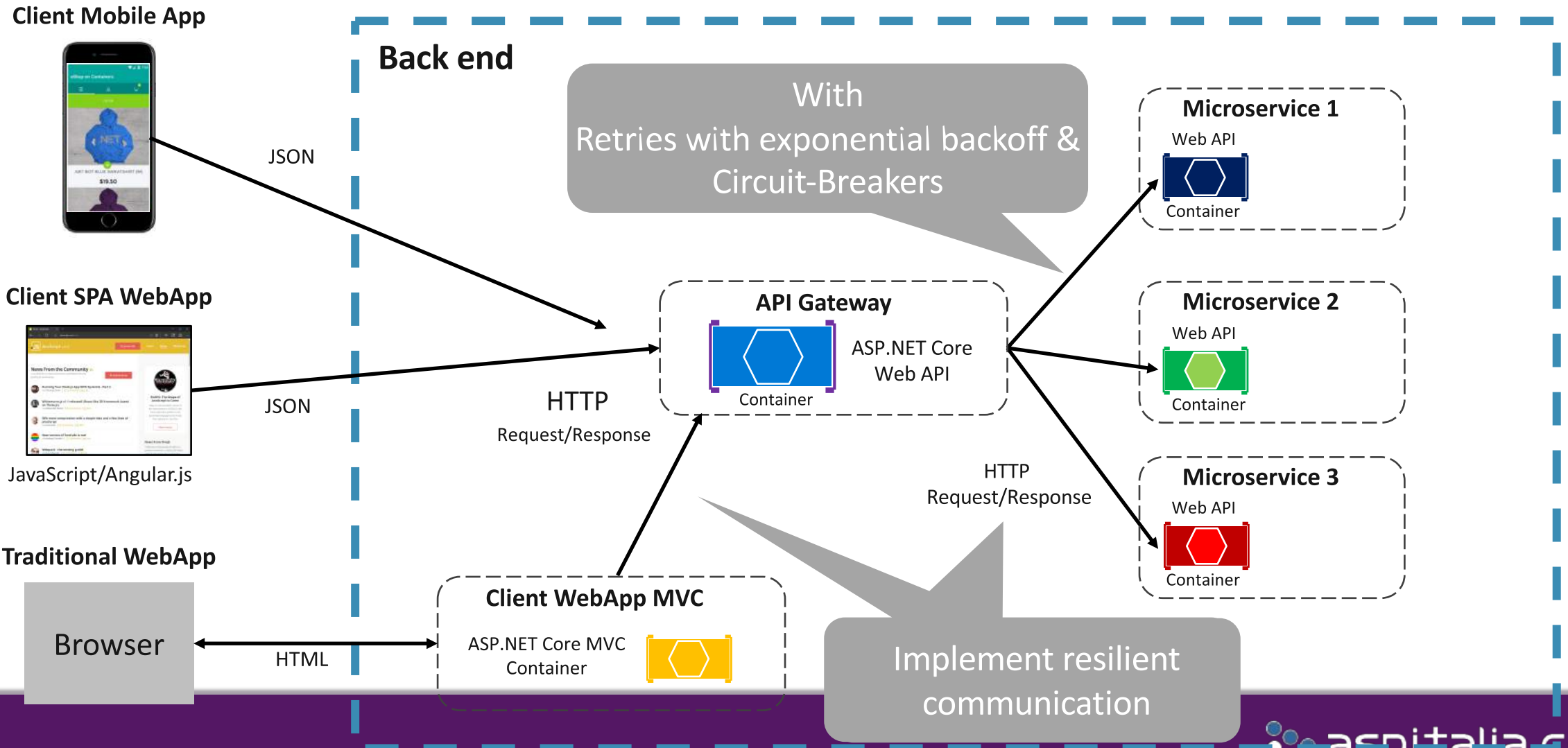
- API Gateways implementation with *Ocelot*

Orchestrators: Scale-out & devs collaboration

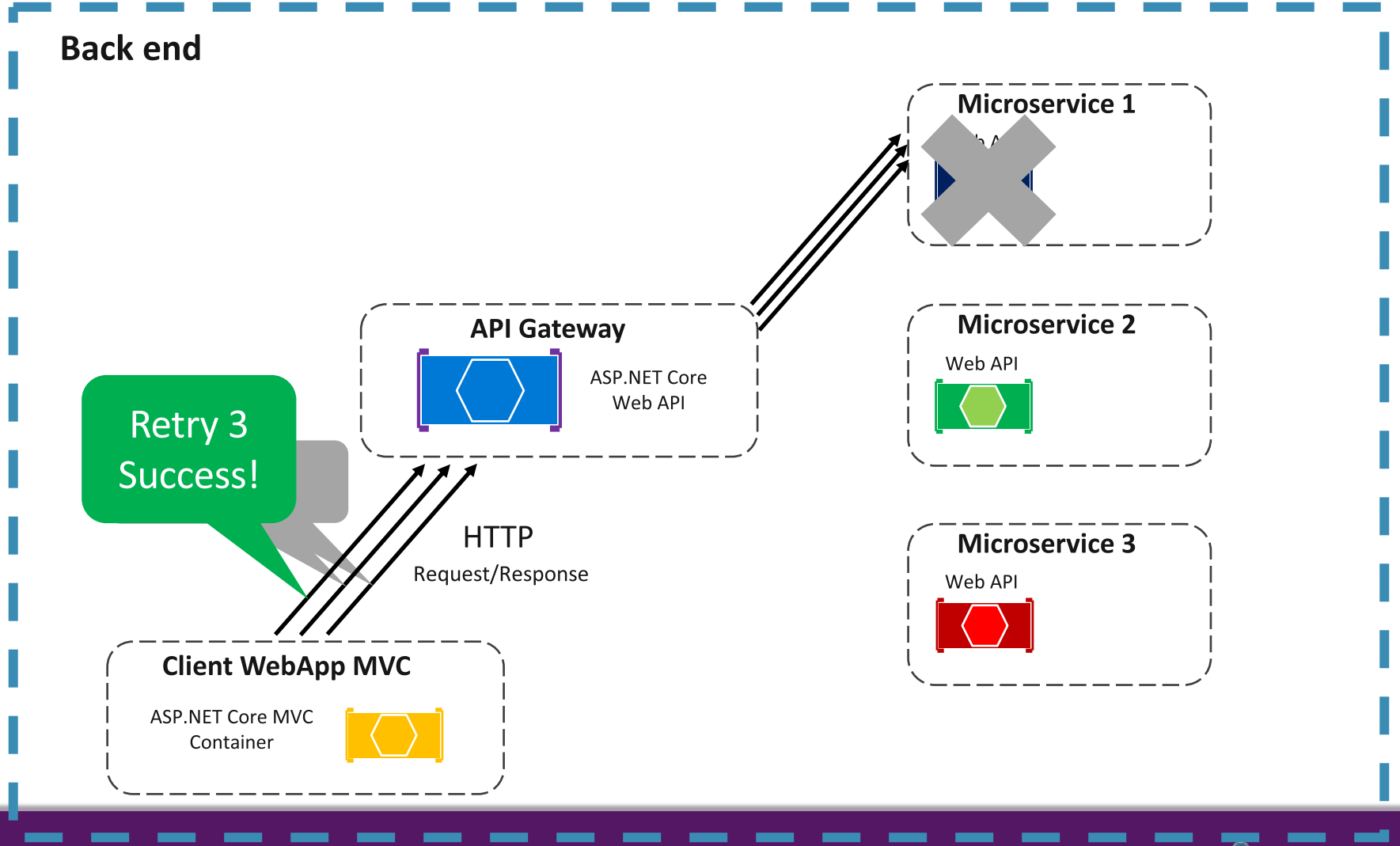
- AKS, Azure *Dev Spaces*, etc.

…and many, many others…

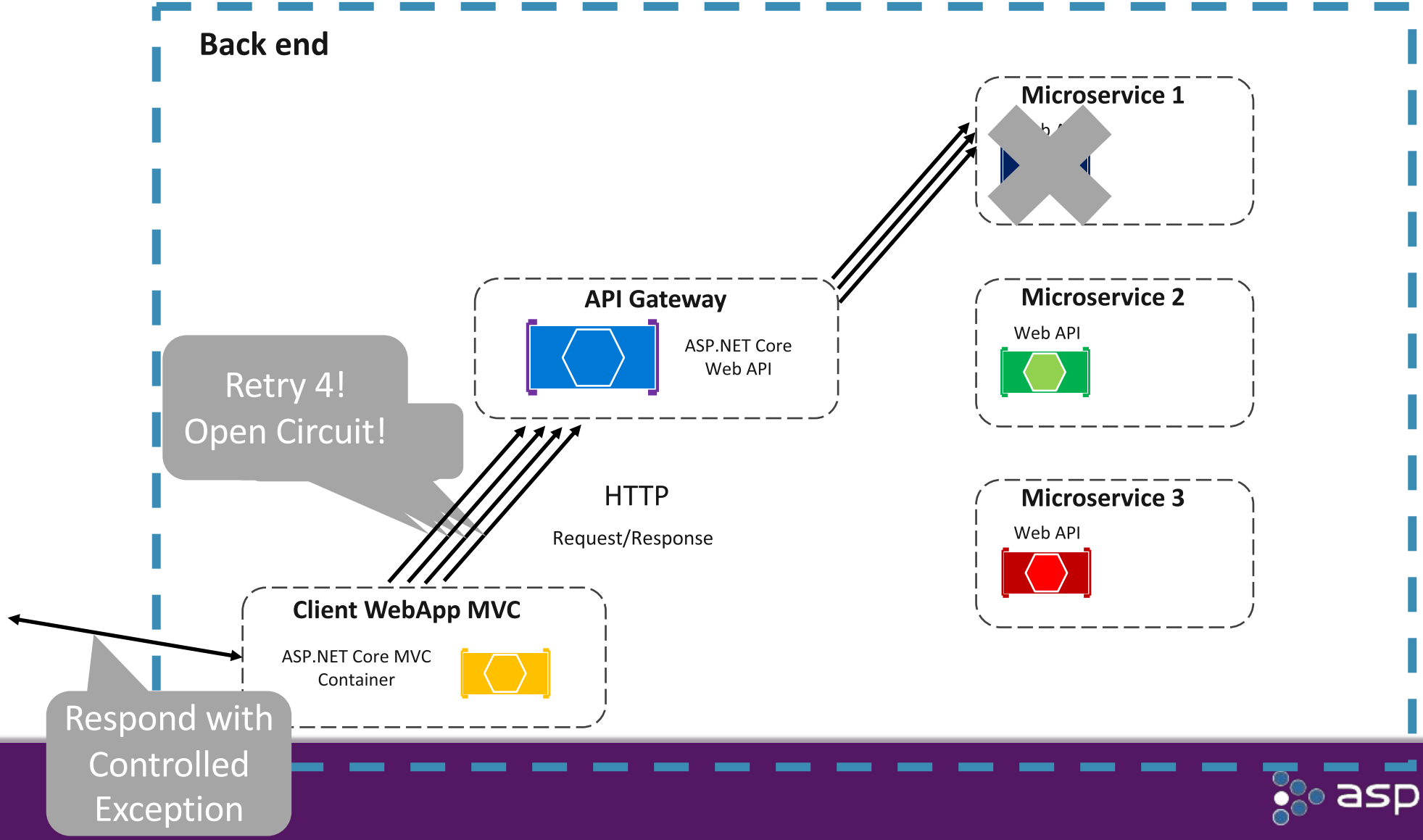
# Resilient cloud applications



# Retry mechanisms



# Retry mechanisms with exponential backoff



# Resilient HTTP requests

**.NET Core 2.1**
**HttpClientFactory**


+

**Policies**

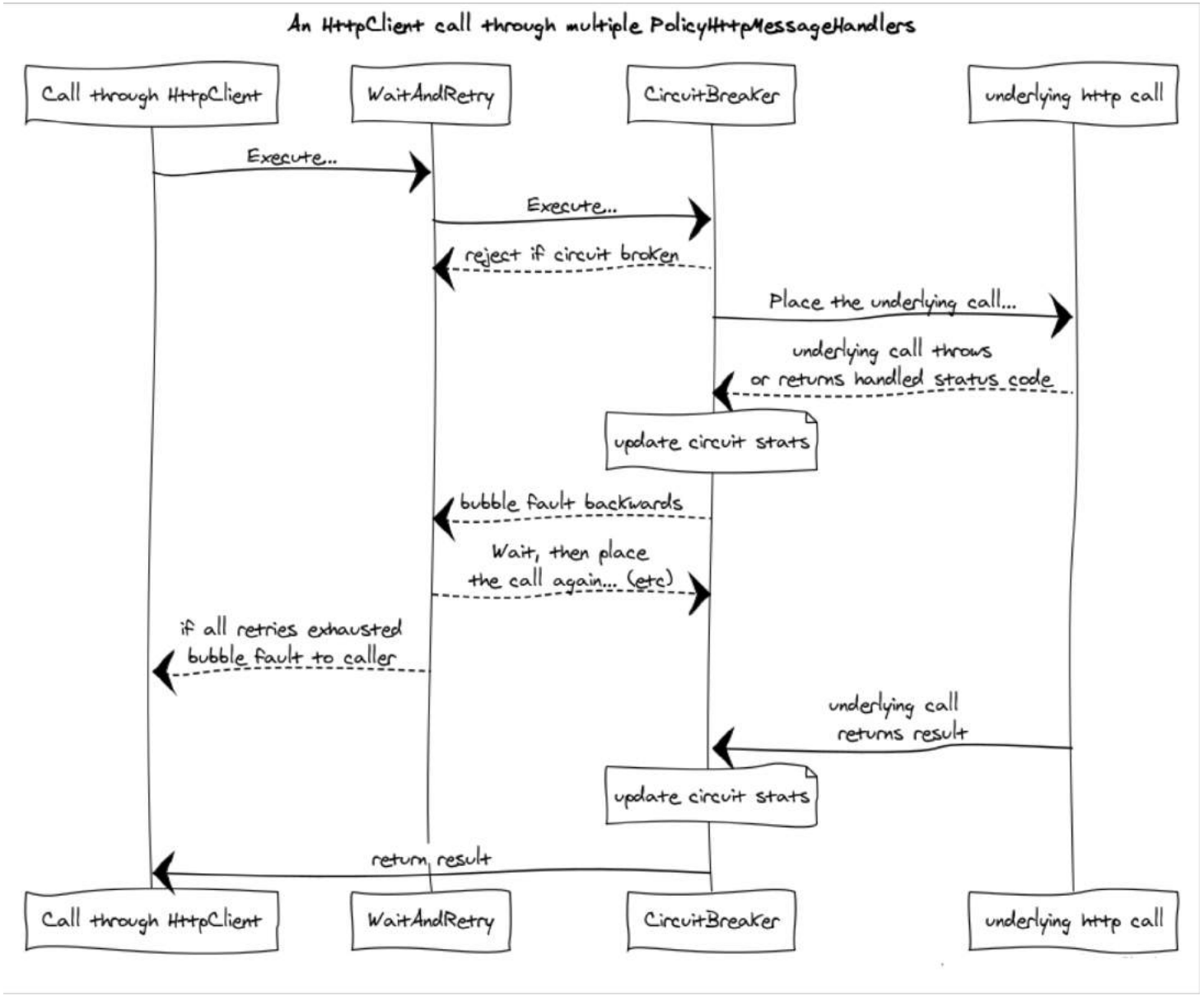
# Polly policies

.NET resilience and transient-fault-handling library

Supports retry, circuit breaker, timeout, bulkhead isolation, and fallback

Thread safe

.NET Standard 1.1 and 2.0+



# HttpClientFactory

## Central location

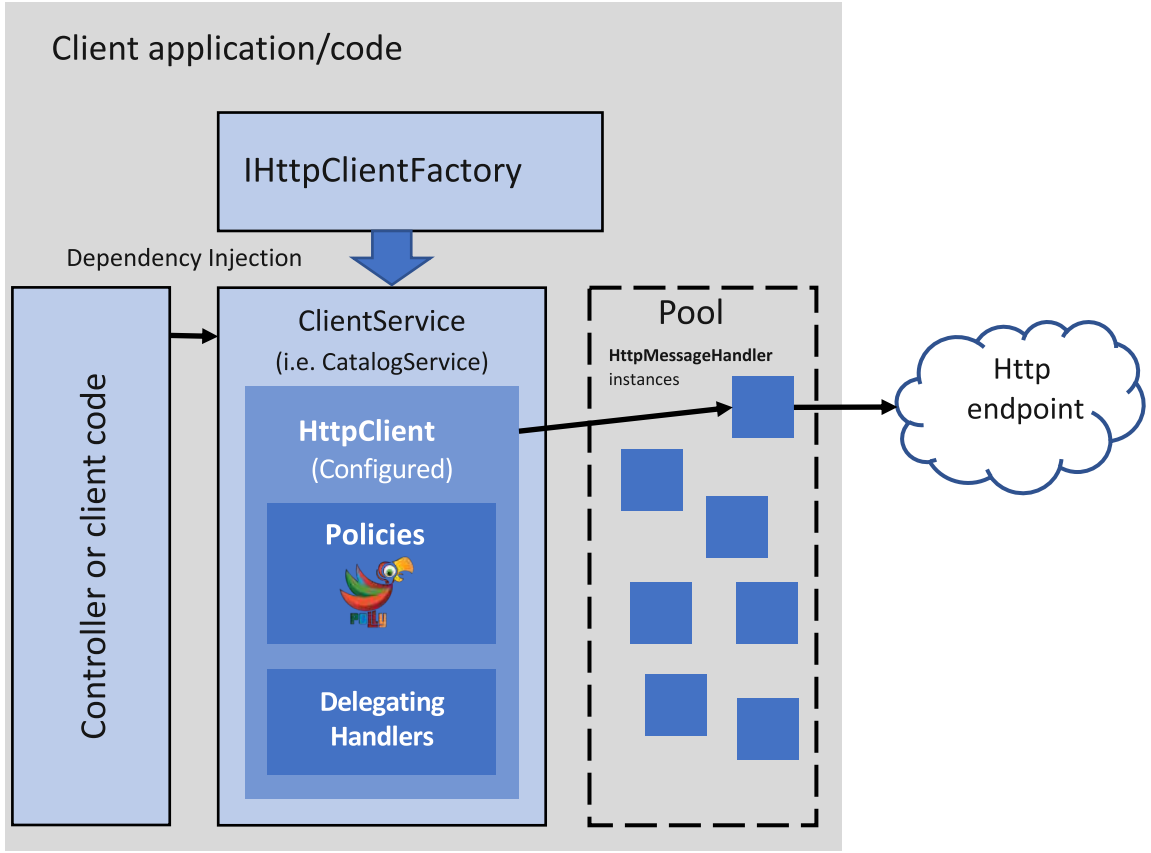
- For configuring typed HTTP clients (Service Agents) or named HTTP clients
- For logging

## Delegating handlers in *HttpClient* such as:

- Resilient HTTP policies (Polly)
- Auth token propagation
- Validations, etc.

## HttpClient lifetimes

- Manage lifetime of *HttpMessageHandler* instances properly (pool of handlers)

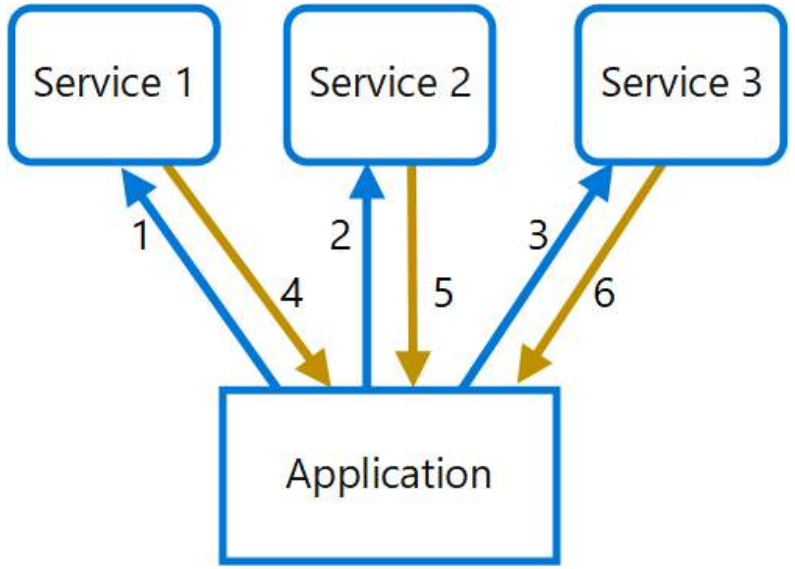


# Demo

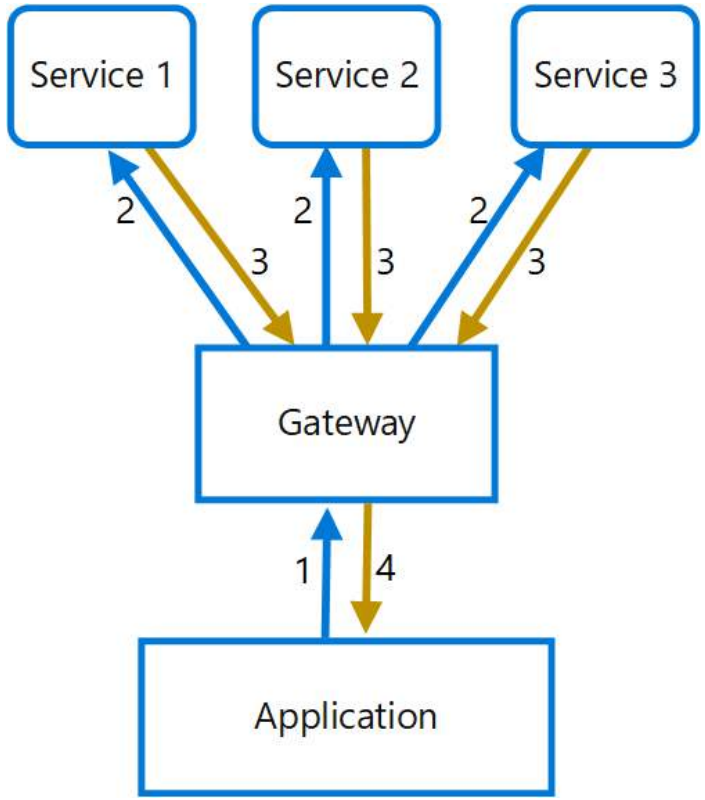
Resilient HTTP calls with *HttpClientFactory* and Polly policies
with retries using exponential backoff and circuit breaker

# Direct communication vs API Gateway

**Direct communication to microservices**
("NO API Gateway usage")

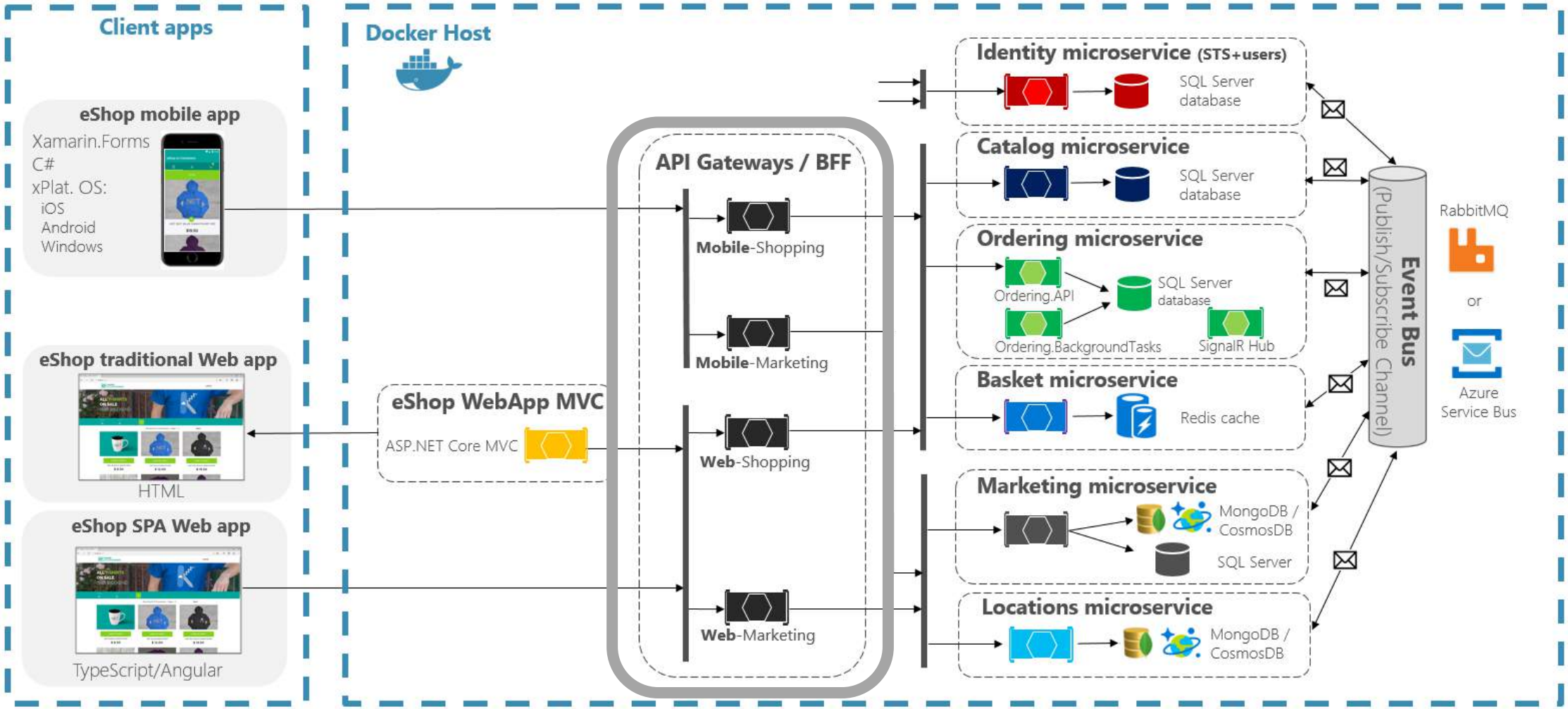



**Using the API Gateway pattern**



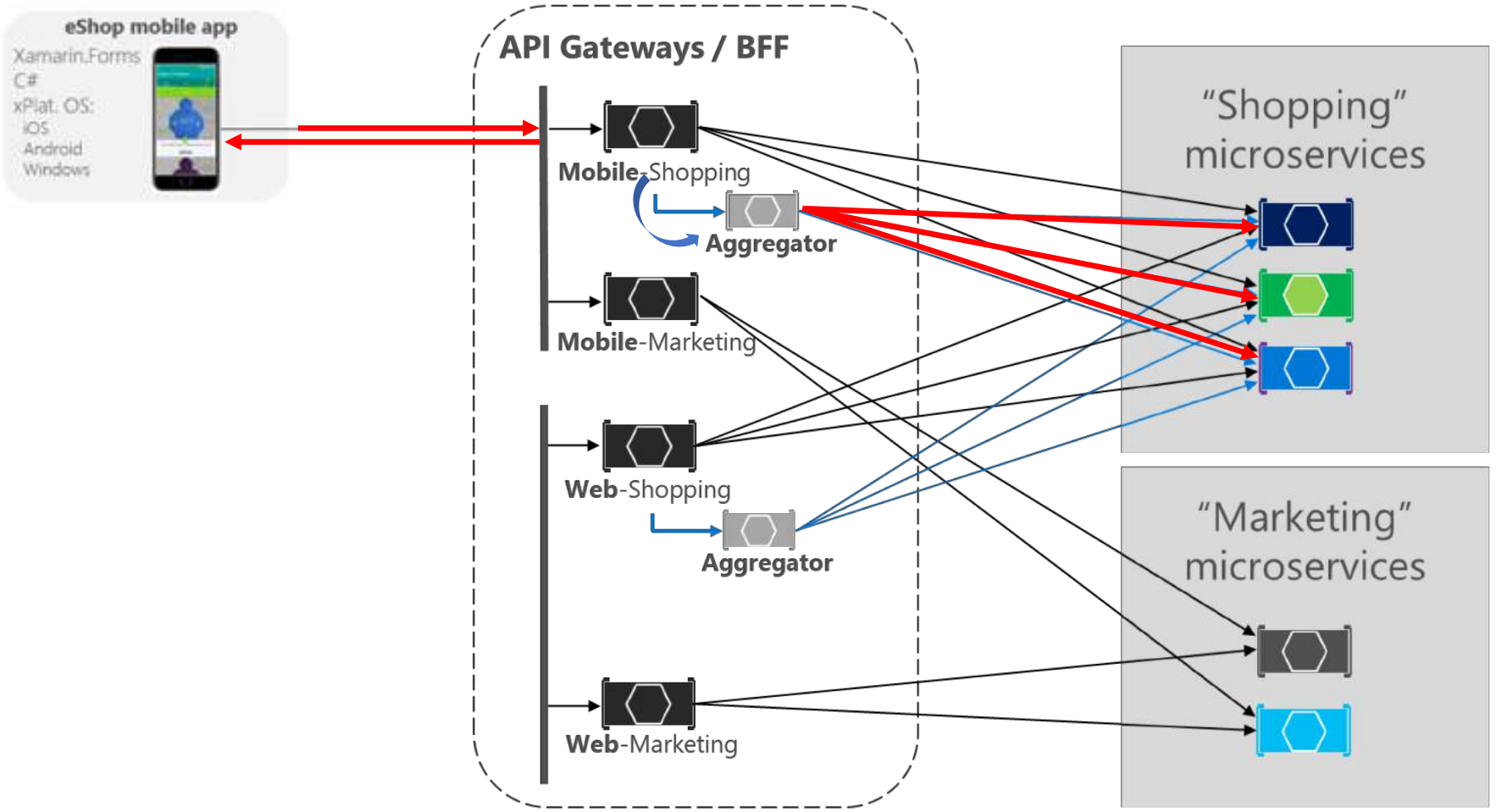
# eShopOnContainers reference application

(Development environment architecture)

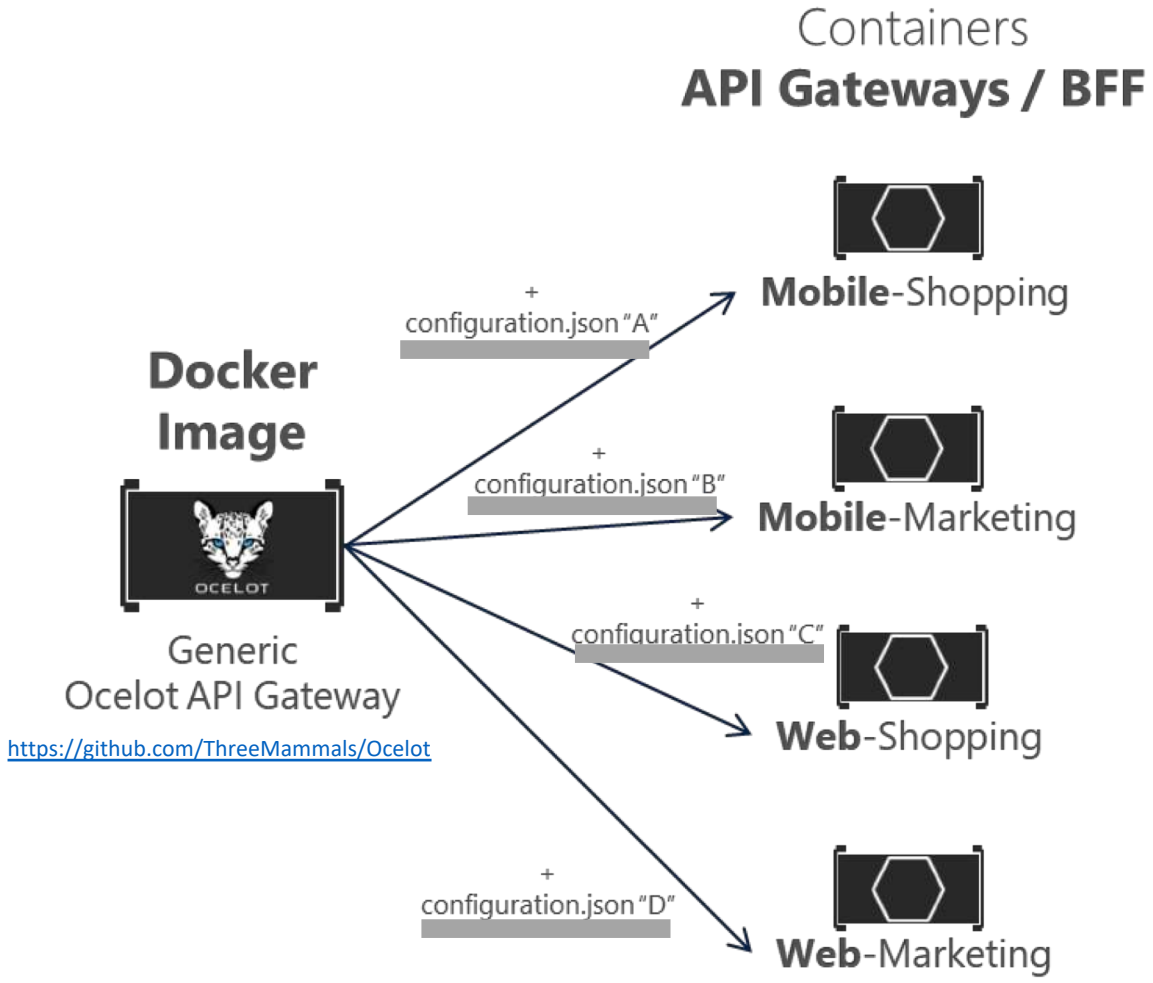


# eShopOnContainers

(API Gateways / BFF and Aggregator-services zoom-in)



# Ocelot API Gateways



# Demo

API-Gateways demo
Ocelot code overview

Dev environment

Production environment

eSHOP onCONTAINERS

Customized

**eShopOnContainers on Azure**

Foundational Development technologies

Cloud infrastructure and Specific Orchestrators

Development

Microsoft .NET

.NET Core
ASP.NET Core

Azure Kubernetes Service (AKS)

Azure Service Fabric

Orchestrators

Deployment

Linux Containers
Windows Containers

Service Bus

SQL Database

BLOB Storage

Cosmos DB

Redis Cache

Key Vault

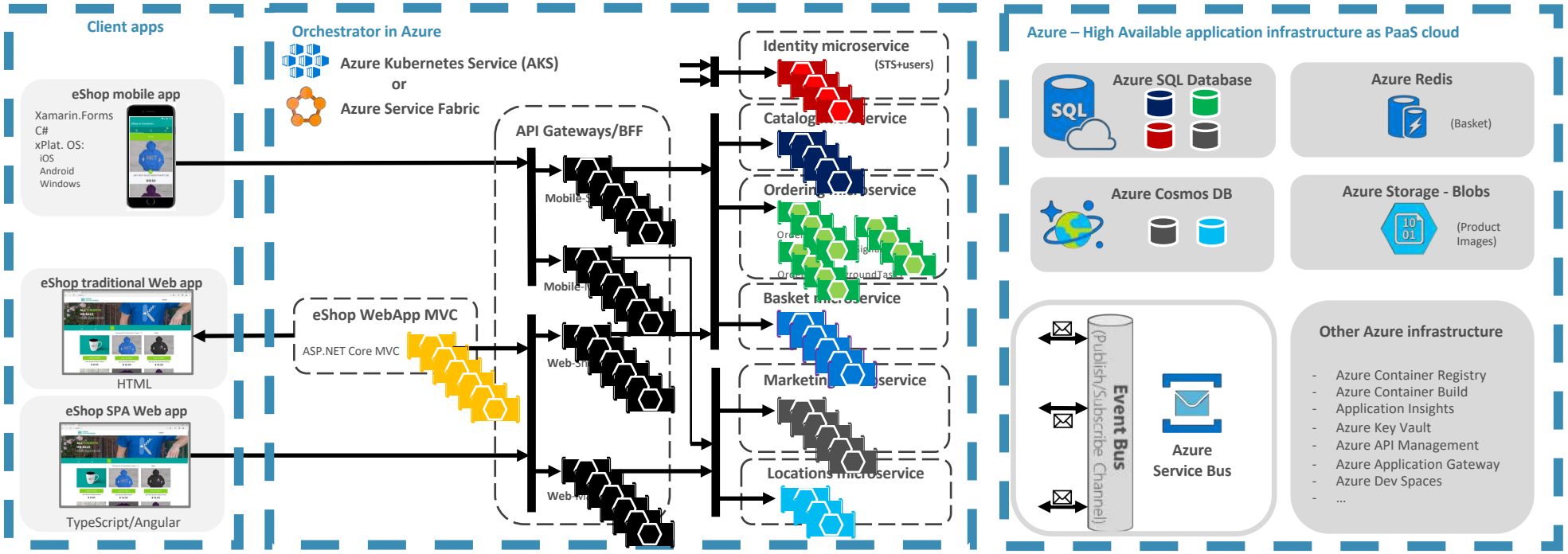
Other Cloud Infrastructure

Exploring Microservices Architecture/Design/Development

Infrastructure Decisions

Production-Ready Microservices

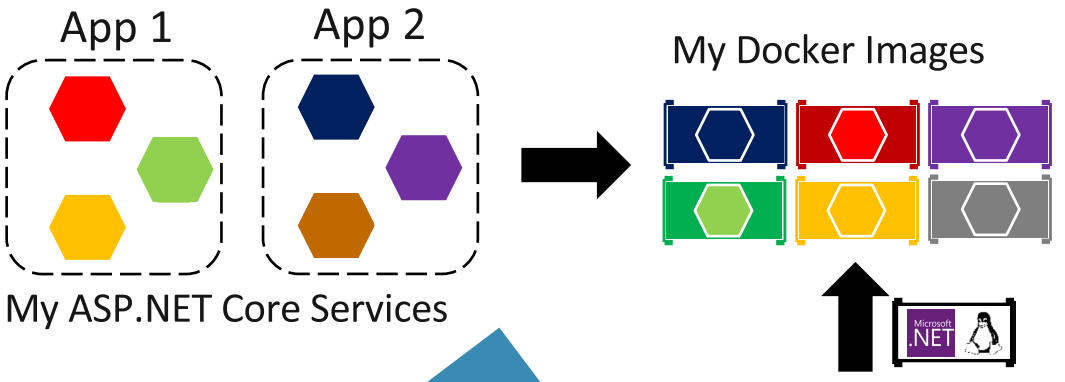
# Sometimes we go live in production…

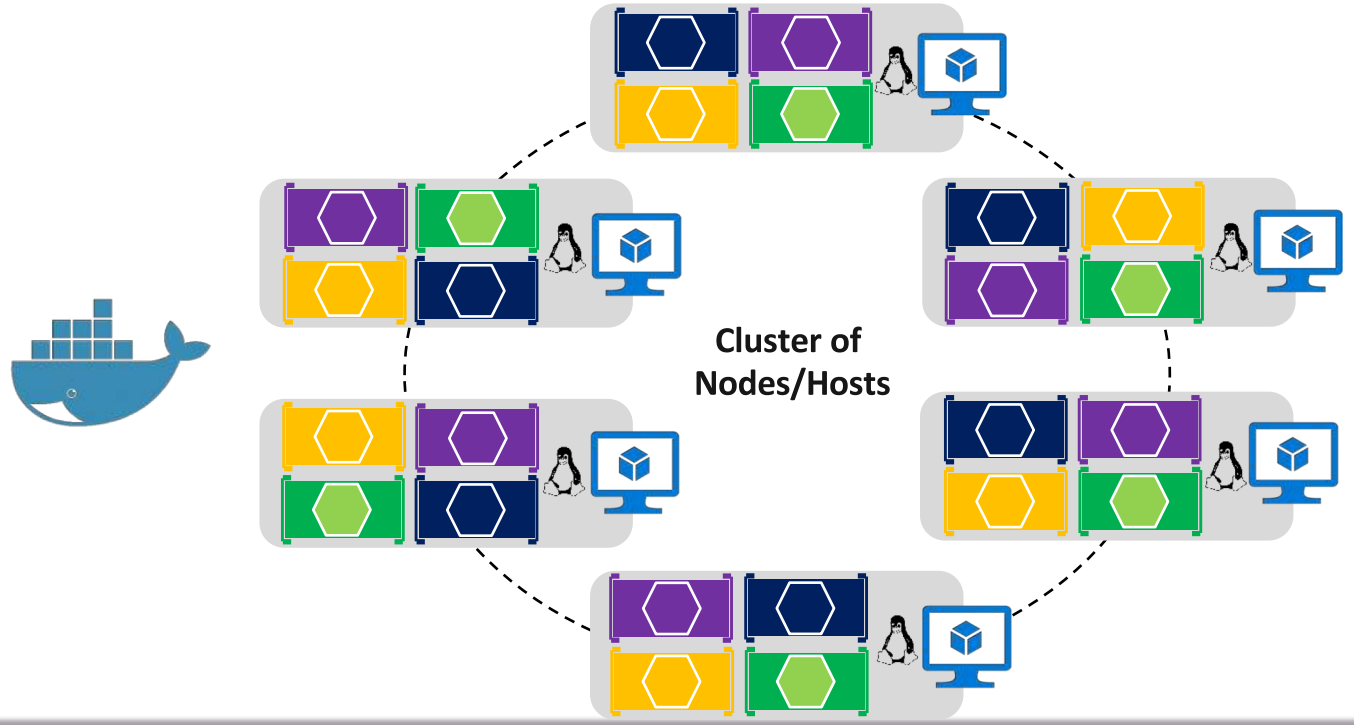


# Cluster managing microservices

Clusters provide:

- High scalability
- Automatic High Availability and resiliency
- High services density per host

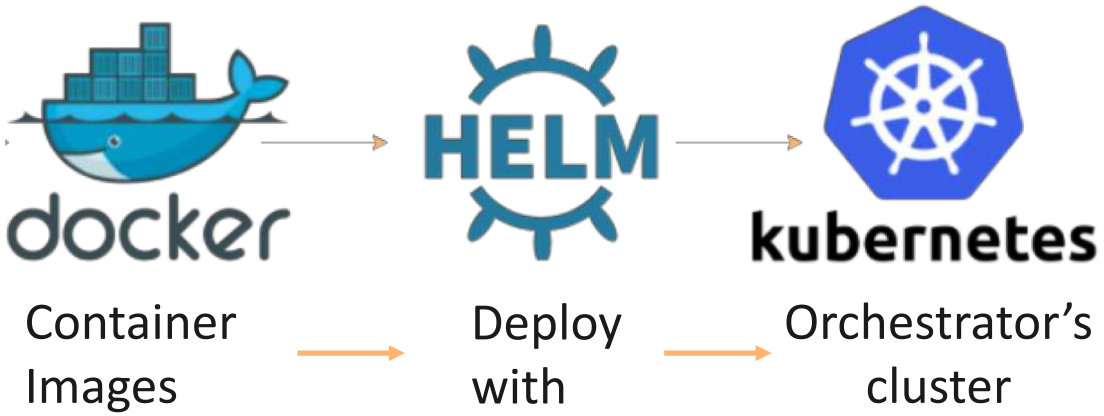


# Choose the right orchestrator

| Azure | Orchestrator / Engine | Description | Good for | Common workloads |
|---|---|---|---|---|
| **Azure Kubernetes Service (AKS)**  | **Kubernetes**  | *Kubernetes* is an open-source platform for automating deployment, scaling, and operations of application containers across clusters of hosts.<br>**AKS: You pay for VMs in cluster**<br>**ACS Engine: IaaS container infrastructure** | It's an OSS ecosystem<br>More mature:  <br>Less mature:  | Microservices based on containers |
| **Azure Service Fabric** (Mesh and cluster)  | **Service Fabric**  | *Azure Service Fabric* is a distributed systems platform that makes it easy to package, deploy, and manage scalable and reliable microservices.<br>**Mesh: PaaS/Serverless platform**<br>**Cluster: You pay for VMs in cluster** | It's a Microsoft ecosystem and OSS<br>More mature:  <br>Less mature:  | a) Microservices based on containers<br>b) Microservices based on plain processes<br>c) Stateful services |

# Kubernetes and HELM



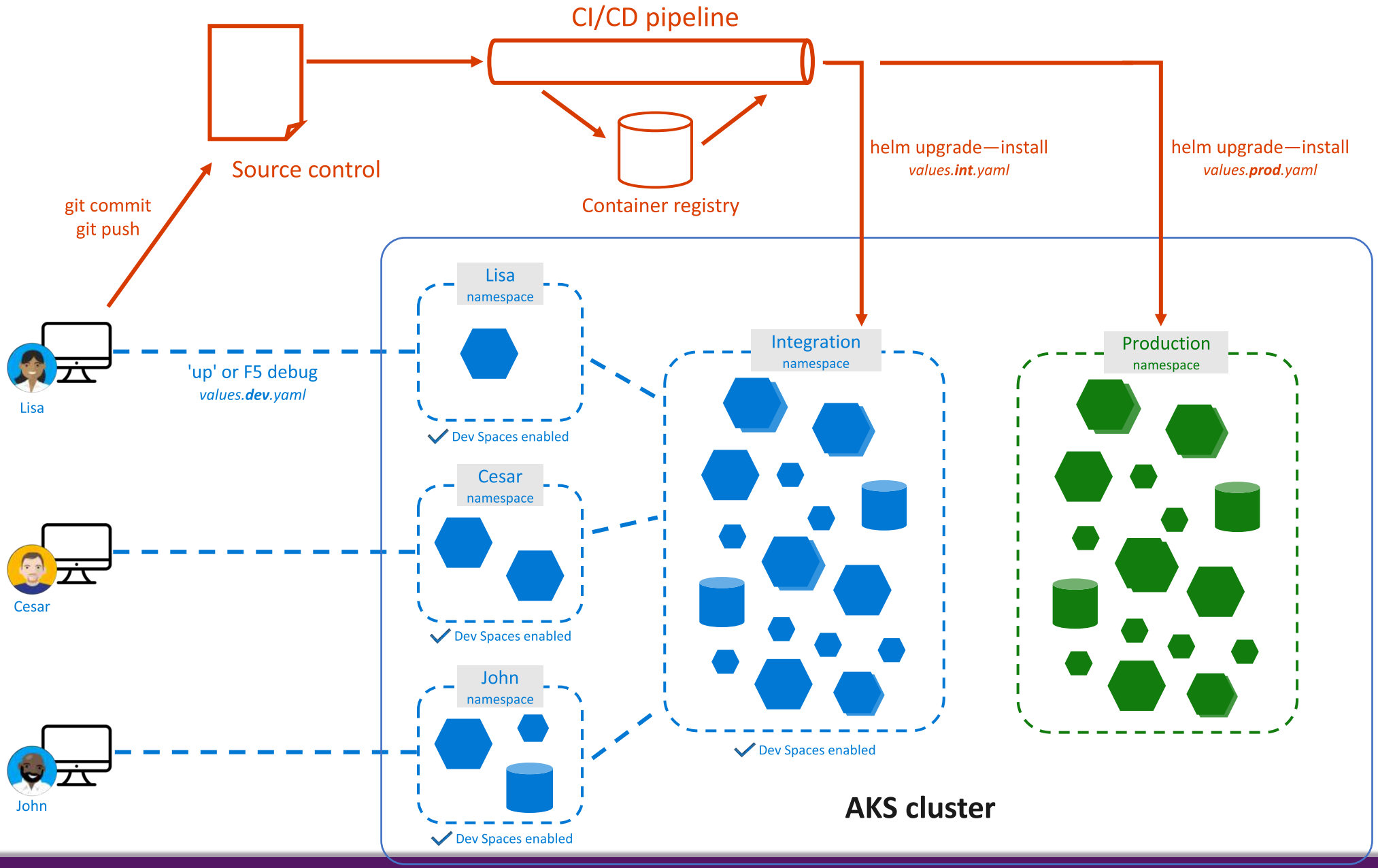

## Helm is THE package manager for Kubernetes:

…Kubernetes deployments with just kubectl and .yml files are not standard but custom & complex…

- Makes application deployment easy, standard and reusable
- Easy application *install*, *update*, *rollback* & *removal*. Packages are declaratively defined in Helm Charts
- Charts can be shared and publicly published (*https://github.com/helm/charts/tree/master/stable*)
- Designed with versioning of packages in mind

# Demo

# ALM


CI/CD pipeline
Source control
Container registry
git commit
git push
helm upgrade—install
values.int.yaml
helm upgrade—install
values.prod.yaml
Lisa
'up' or F5 debug
values.dev.yaml
Cesar
John
Lisa
namespace
Dev Spaces enabled
Cesar
namespace
Dev Spaces enabled
John
namespace
Dev Spaces enabled
Integration
namespace
Dev Spaces enabled
Production
namespace
AKS cluster

# Azure DevOps

## Azure Boards

Deliver value to your users faster using proven agile tools to plan, track, and discuss work across your teams.

## Azure Pipelines

Build, test, and deploy with CI/CD that works with any language, platform, and cloud. Connect to GitHub or any other Git provider and deploy continuously.

## Azure Repos

Get unlimited, cloud-hosted private Git repos and collaborate to build better code with pull requests and advanced file management.

## Azure Test Plans

Test and ship with confidence using manual and exploratory testing tools.

## Azure Artifacts

Create, host, and share packages with your team, and add artifacts to your CI/CD pipelines with a single click.

https://azure.com/devops

# Azure Repos

### Works with your Git client

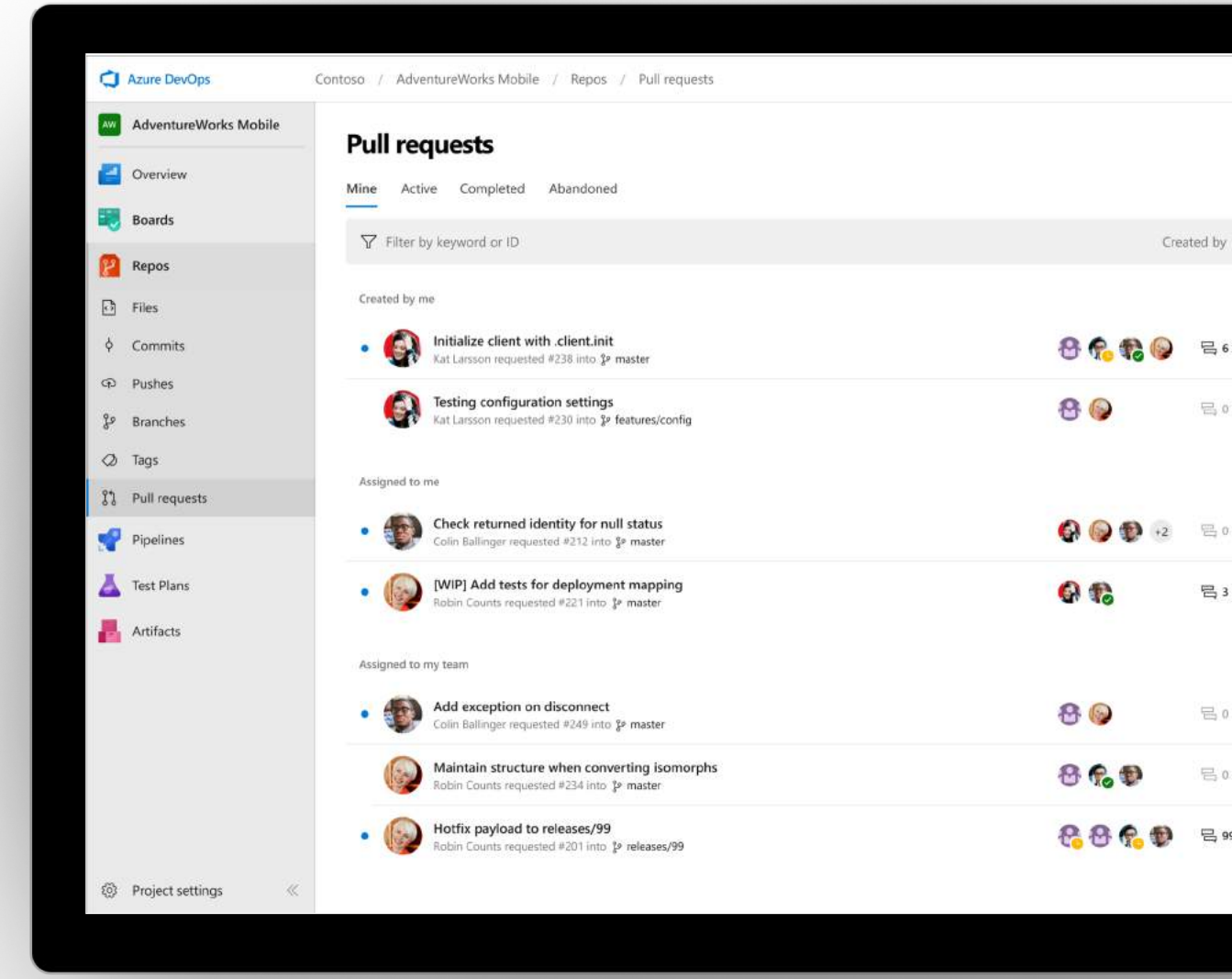
Securely connect with and push code into your Git repos from any IDE, editor, or Git client.

### Web hooks and API integration

Add validations and extensions from the marketplace or build your own using web hooks and REST APIs.

### Semantic code search

Quickly find what you're looking for with code-aware search that understands classes and variables.



# Azure Boards

## Connected from idea to release

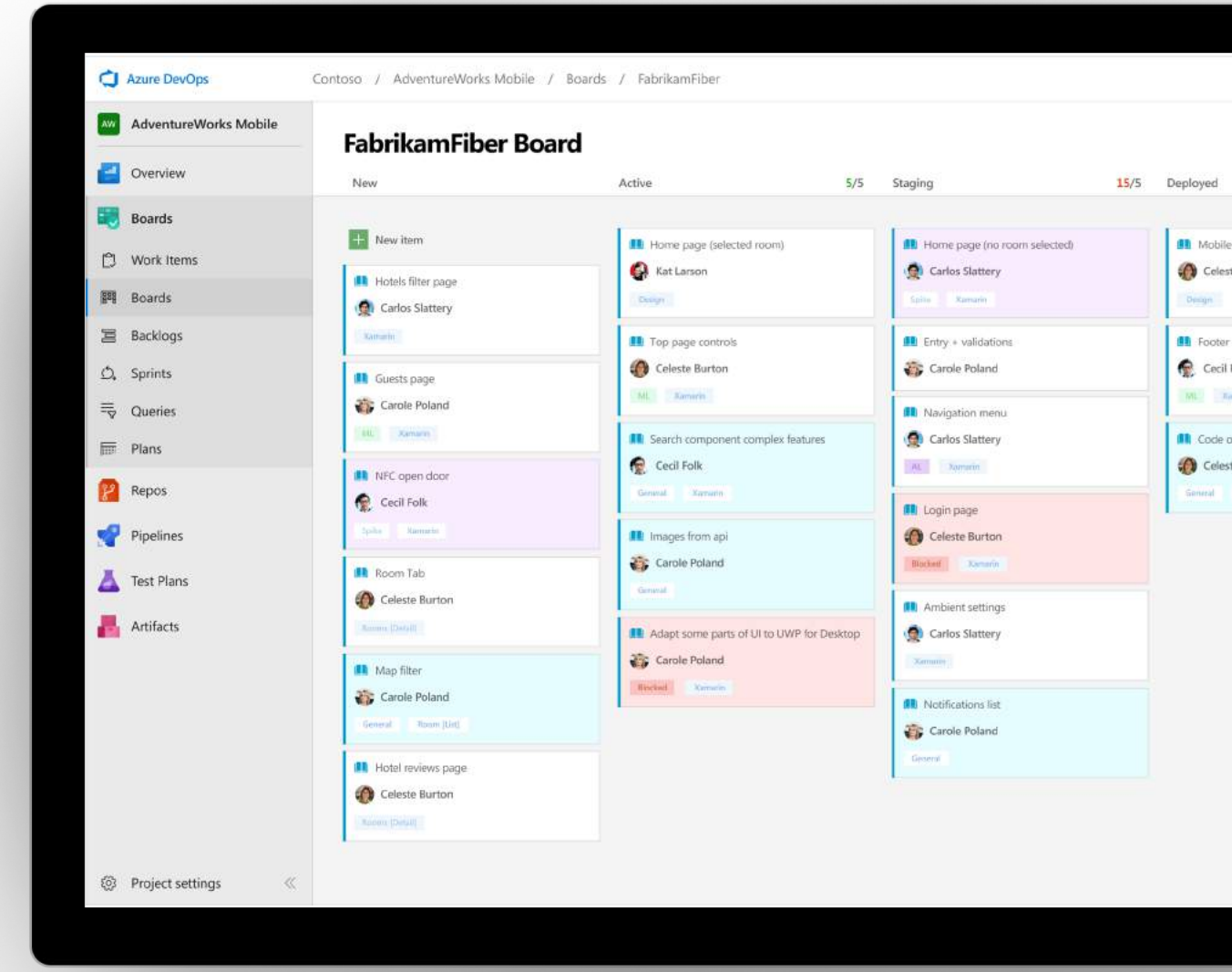
Track all your ideas at every development stage and keep your team aligned with all code changes linked directly to work items.

## Scrum ready

Use built-in scrum boards and planning tools to help your teams run sprints, stand-ups, and planning meetings.

## Project insights

Gain new insights into the health and status of your project with powerful analytics tools and dashboard widgets.



# Azure Pipeline

## Any language, any platform, any cloud

Build, test, and deploy Node.js, Python, Java, PHP, Ruby, C/C++, .NET, Android, and iOS apps. Run in parallel on Linux, macOS, and Windows. Deploy to Azure, AWS, GCP or on-premises

## Extensible

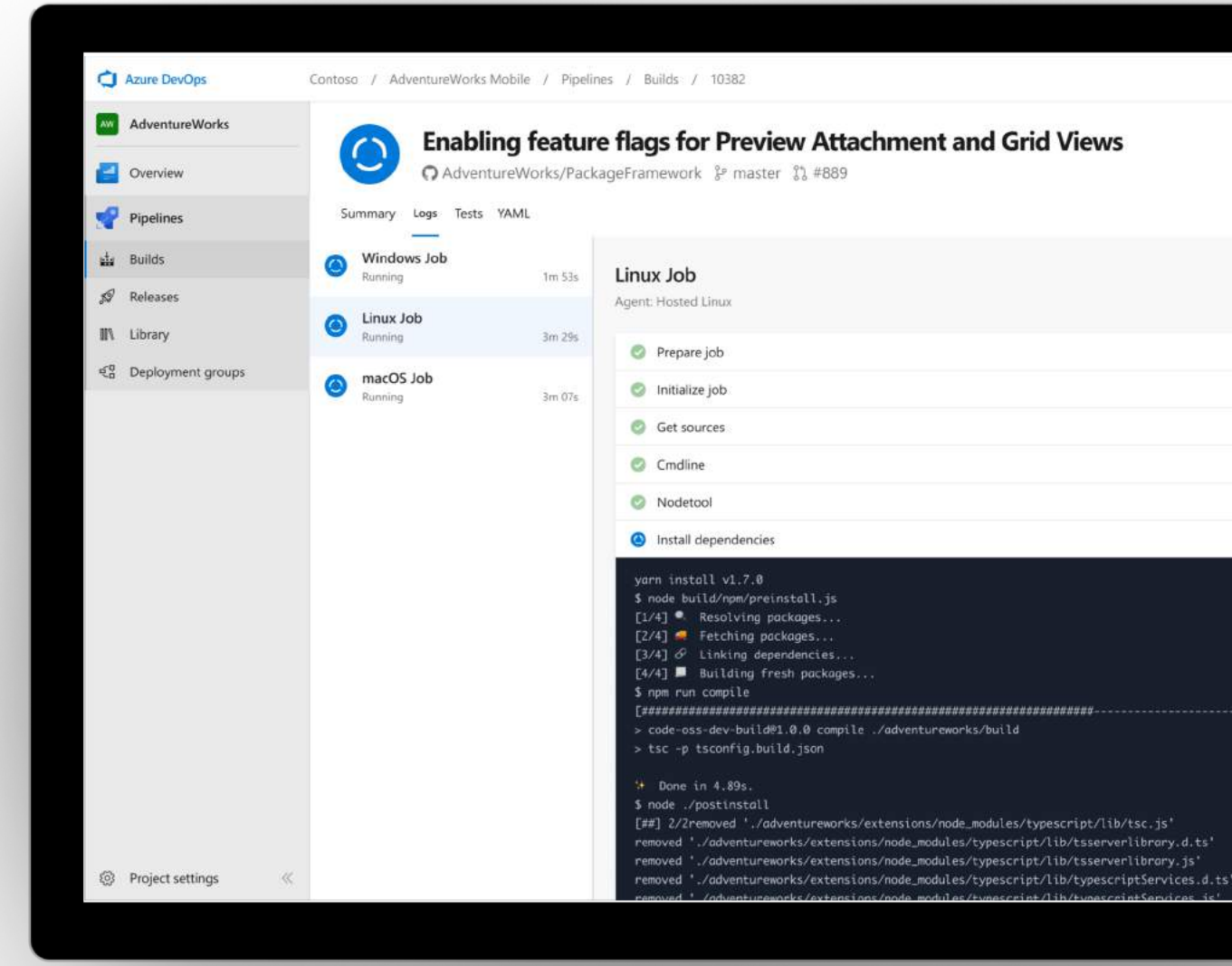
Implement a wide range of community-built build, test, and deployment tasks, along with hundreds of extensions from Slack to SonarCloud. Support for YAML, reporting and more

## Containers and Kubernetes

Easily build and push images to container registries like Docker Hub and Azure Container Registry. Deploy containers to individual hosts or Kubernetes.

## Best-in-class for open source

Ensure fast continuous integration/continuous delivery (CI/CD) pipelines for every open source project. Get unlimited build minutes for all open source projects with up to 10 free parallel jobs across Linux, macOS and Windows



# Azure Test Plans

## Capture rich data

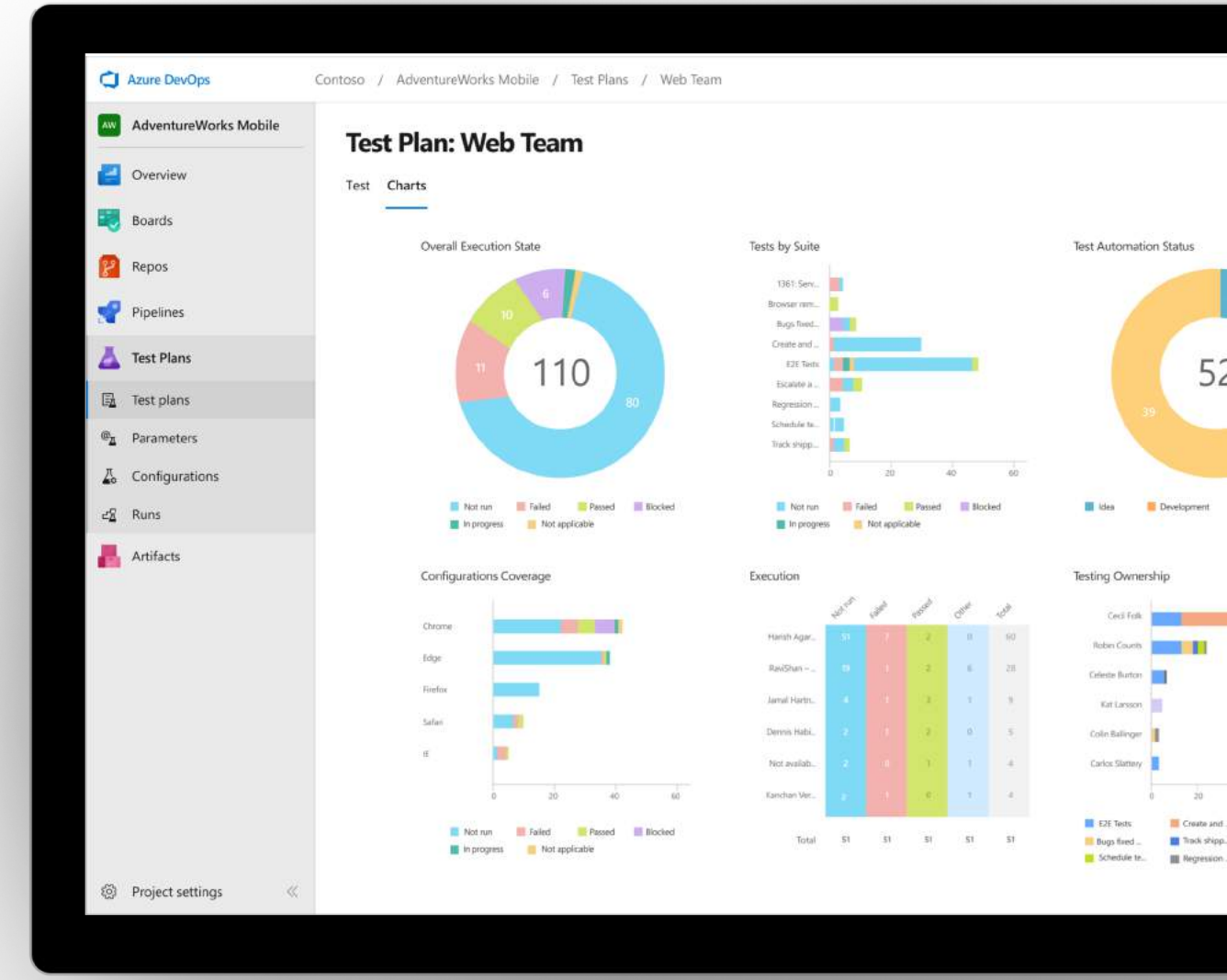
Capture rich scenario data as you execute tests to make discovered defects actionable. Explore user stories without test cases or test steps. You can create test cases directly from your exploratory test sessions.

## Test across web and desktop

Test your application where it lives. Complete scripted tests across desktop or web scenarios. Test on-premises application from the cloud and vice-versa.

## Get end-to-end traceability

Leverage the same test tools across your engineers and user acceptance testing stakeholders. Pay for the tools only when you need them.



# Azure Artifacts

### Manage all package types

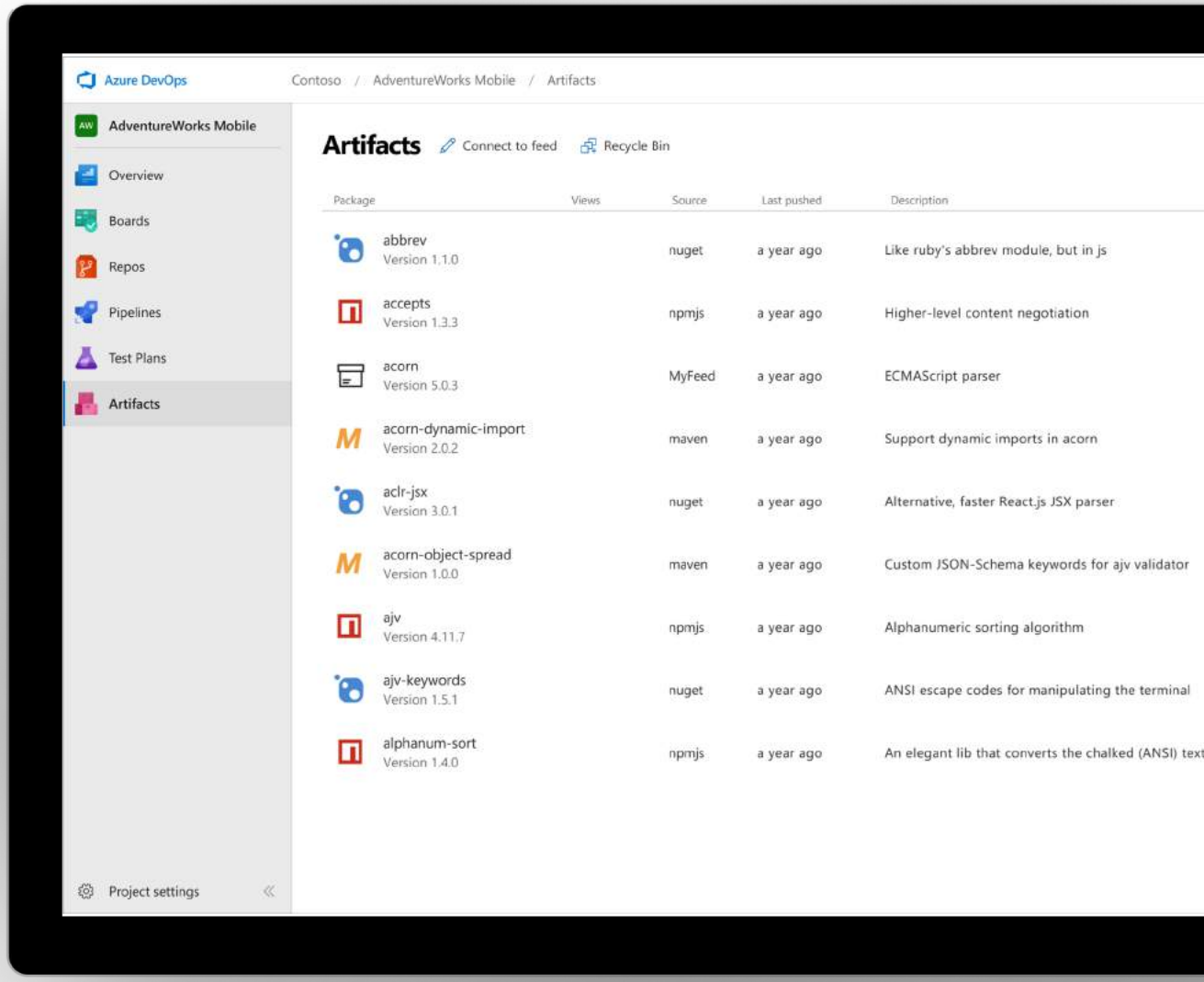
Get universal artifact management for Maven, npm, and NuGet.

### Add packages to any pipeline

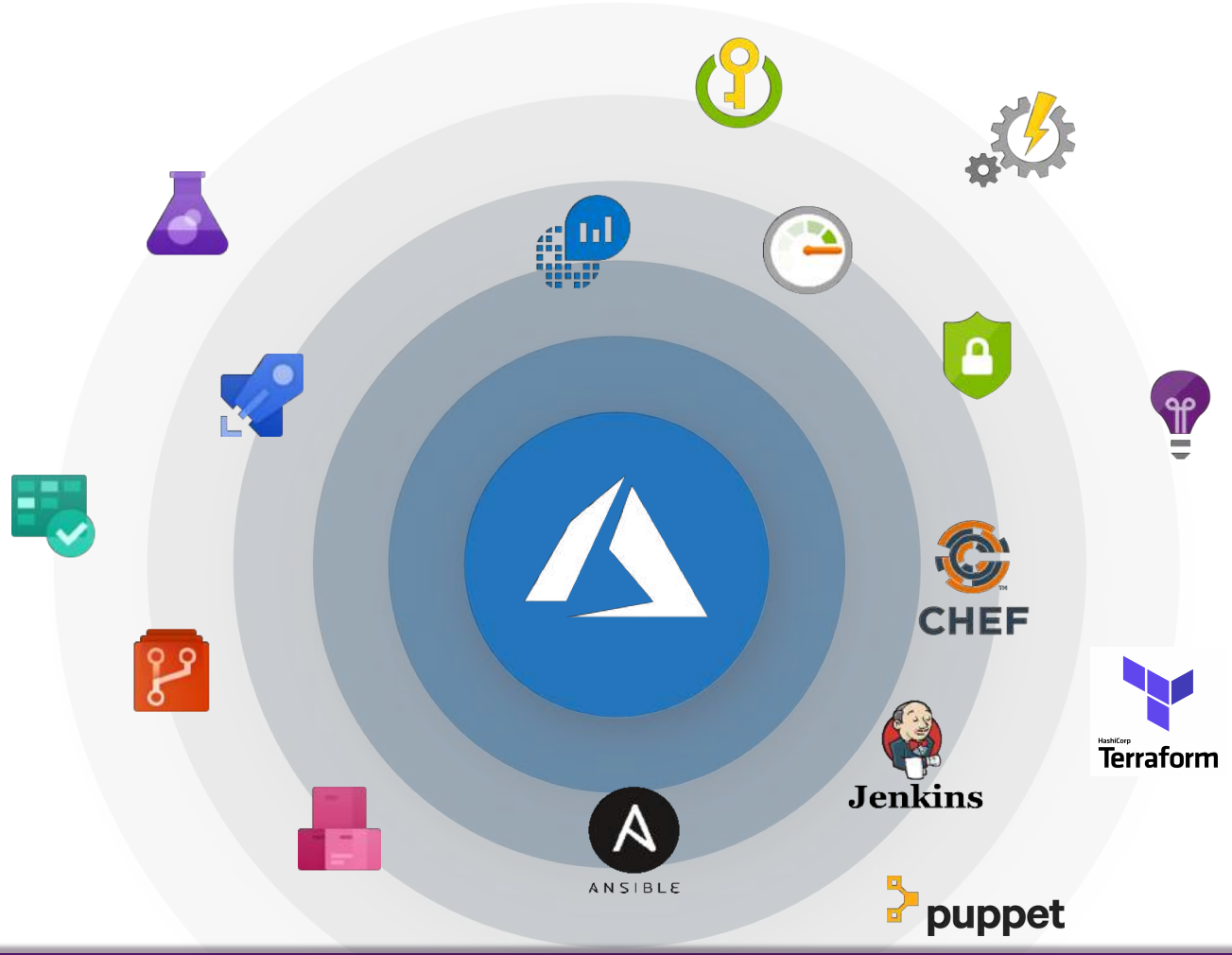
Share packages, and use built-in CI/CD, versioning, and testing.

### Share code efficiently

Easily share code across small teams and large enterprises.



# Increasing the Azure ecosystem



# Infrastructure as code

Infrastructure as Code, built-in

Azure Config & Automation

Support for 3rd party and OSS tooling such as Terraform, Ansible, Chef, Puppet & SaltStack

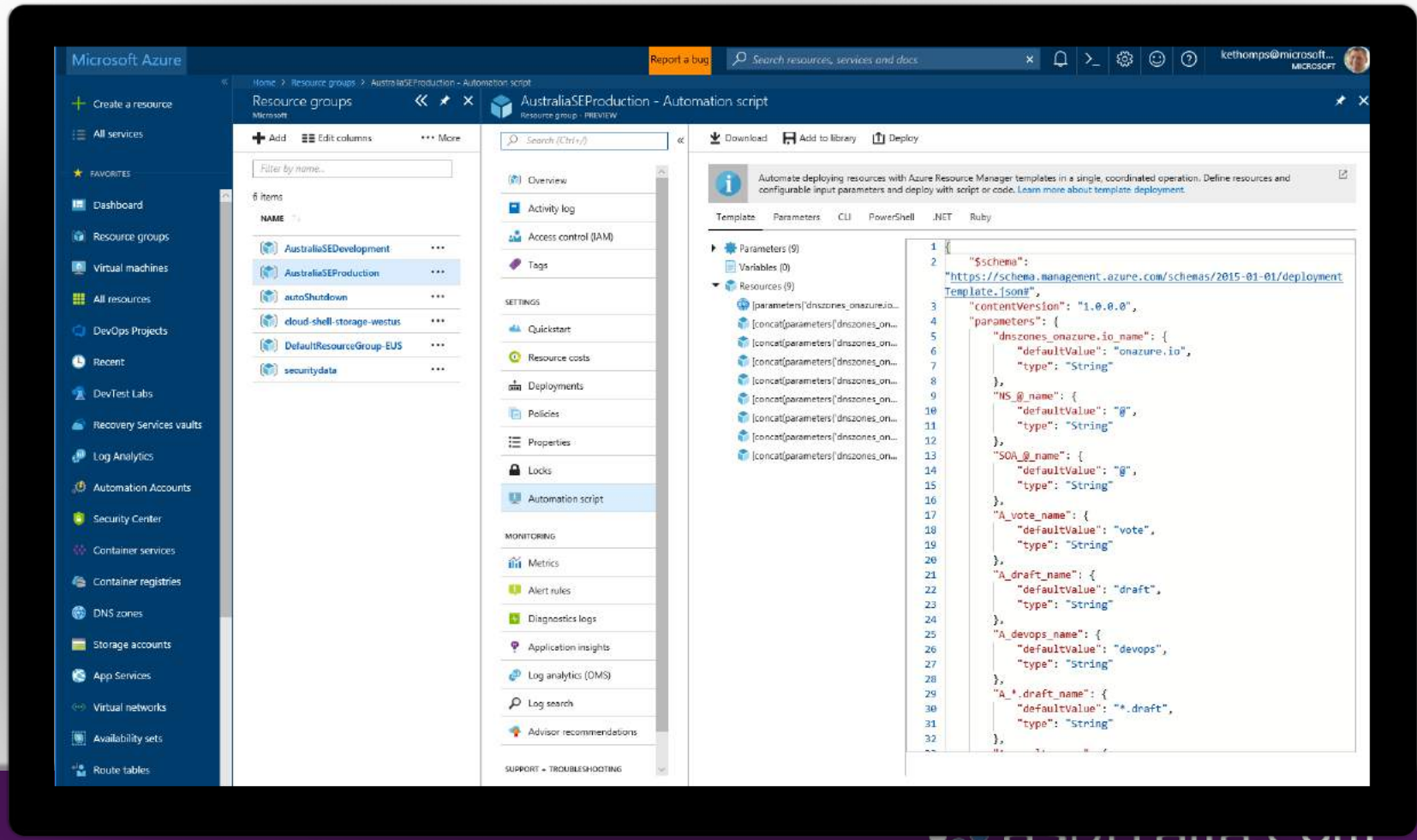
TERRAFORM

ANSIBLE

CHEF

puppet

SALTSTACK



# Changes from VSTS/TFS

## What's new

Existing *https://contoso.visualstudio.com* URL continues to work. ***https://dev.azure.com/contoso*** available for opt-in.

New UI opt-in per user as preview feature. Will start advertising new UI once feedback from new accounts and early adopters has been incorporated.

Can disable services on a per project basis for new UI

New Azure branding in communications and documentation

**TFS will remain the on-premises brand** until the next major version in 2019. The new UI will be enabled in that release.

Existing TFS branded information and downloads remain in Visual Studio locations until next release.

## Pricing

Pipelines can be used independently from Repos — so if you are only using Pipelines and your repos are hosted on GitHub you don't need to pay for Repos or Boards (Basic) users.

The free tier for Pipelines now includes 1,800 minutes per month, up from 240.

**Public project usage is now free.**

# Benefits of a cloud-native solution

Global availability

Hosted and maintained by Microsoft with 99.9% uptime guarantee and 24x7 support

Immediate access to latest features

Simplified deployment to Azure

# Security

Strong integration with Active Directory (or AAD)

- Also works on cloud with on-prem AD through AD Connect
- Only members inside AD can join a VSTS account / team project (external guest access may be enabled inside AD)

Multi factor authentication

Conditional access

- Access may be granted through conditional access policies (based on group membership, device, location and so on…)

# Security

User provisioning and de-provisioning

- Provisioning is automatic thanks to AAD
- Users gain access almost immediatly

Access management

- Permissions based on AD groups

Account and team project policies

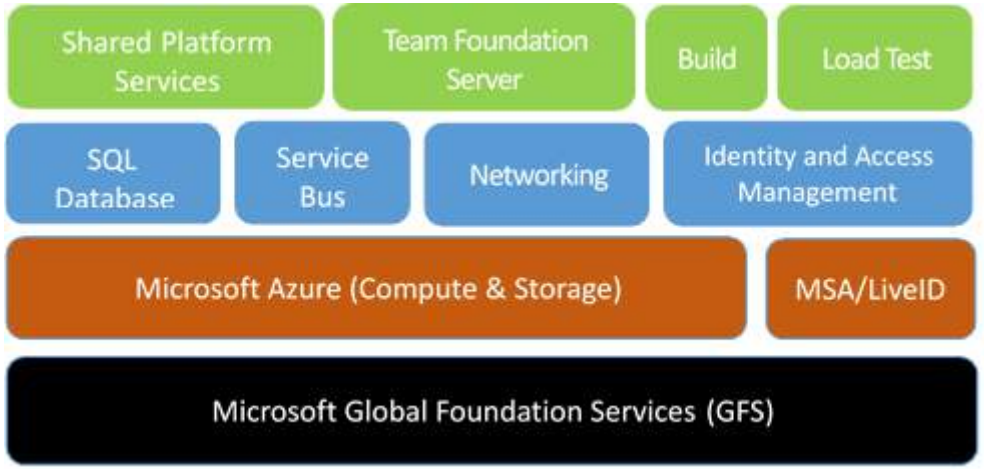
- Granular policies for branch, build, release, package and so on for each user…

# Data availability

VSTS is built on Microsoft Azure platform

Data geo-replica in opposite DCs within the same region

Daily backup of the entire system



# Migration from VSTS/TFS


## TFS Import Service

- Fully supported high fidelity migration path
- Faster and easier to use
- Trusted by enterprises

https://aka.ms/tfsimport

# Demo

# Recap

- Microservices allow to evolve, deploy and scale parts of the application independently
- Microservices offer great benefits but also new challenges (distributed software challenges)
- Microservices are not suitable for all apps, but for large, scalable and long-term evolving applications with typically multiple autonomous development teams
- But you can start extending your existing monolithic apps with new microservices, step-by-step
- Azure + Azure DevOps (and automation) will help you manage all the services and deploy them to different environments at scale

# Grazie!

**@xtumiox**
**matteot@aspitalia.com**

**Materiale su**
**http://aspit.co/netconf-18**