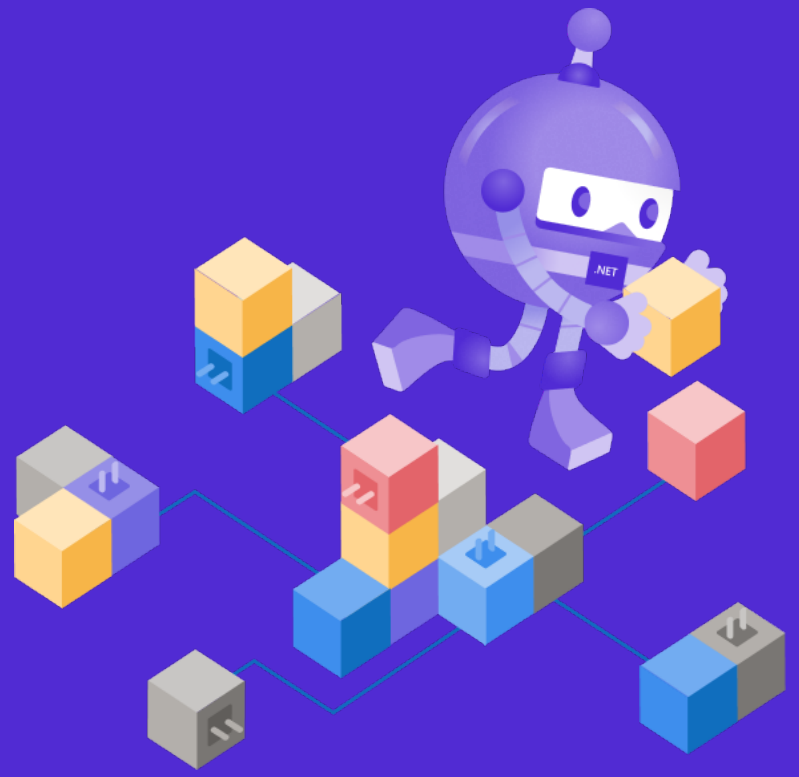
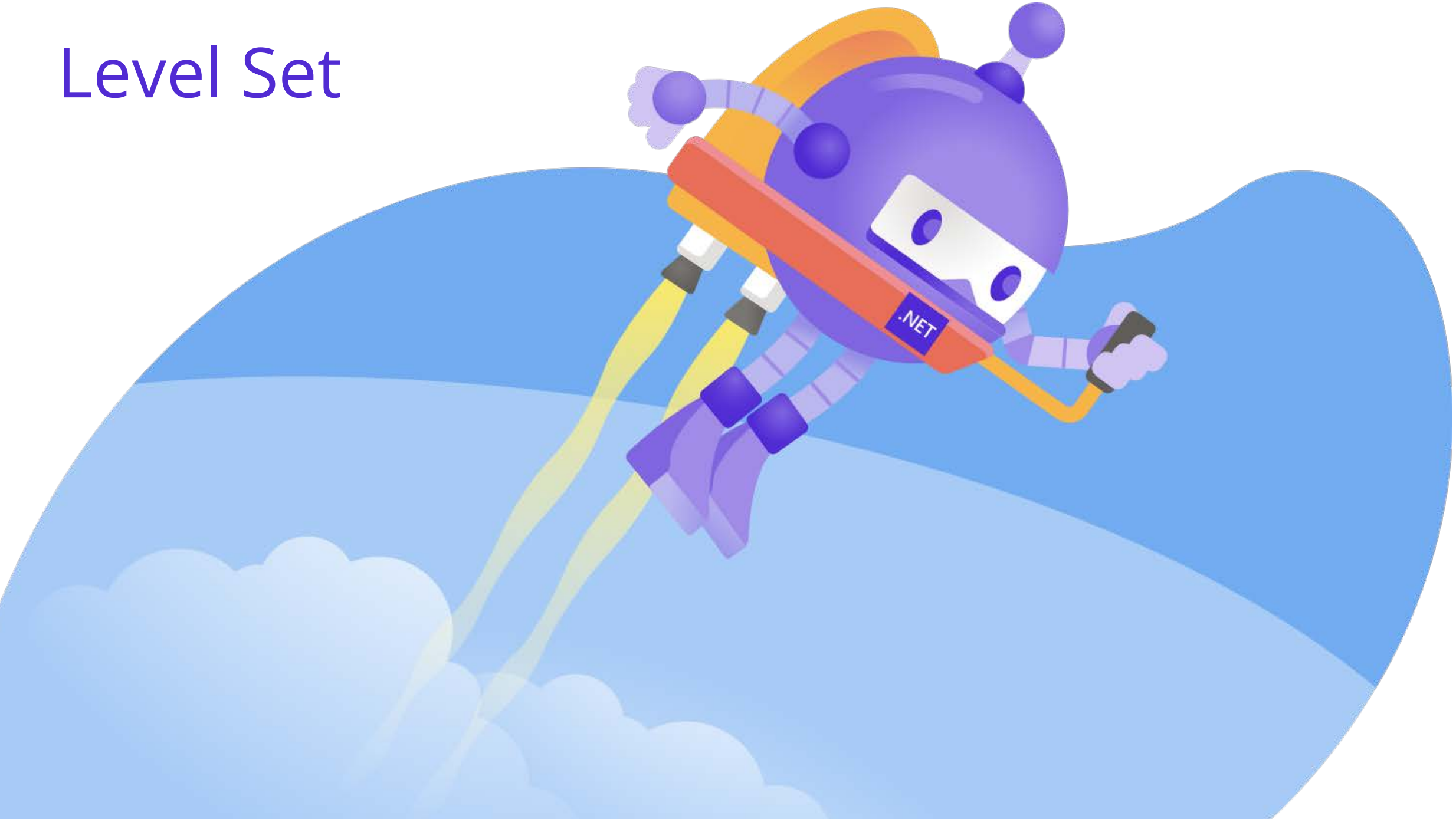
Microservices *On-Ramp*

Rob Vettor

Monu Bambroo



Level Set



What are Microservices???

Ask 10 people, get 7 different answers!

Let's start by agreeing on a ***common definition***

Microservices are…

An approach to system development in which a (larger) app is built as a suite of modular services

What are Microservices?

Each service supports a specific business goal (capability) – a single concern

Move from ➔ *Single Responsibility Principle (SRP)*
To ➔ *Single Concern Principle (SCP)*

What are Microservices?

Each service is fully independent and self-contained, with its own code, data, and state

What are Microservices?

Each service exposes a well-defined interface to communicate with other services

What are Microservices?

This interface is exposed over a platform-agnostic protocol: REST or gRPC

What are Microservices?

Services can be implemented across technology stacks, if desired

What are Microservices?

What are Microservices?

Each can evolve independently and deploy frequently

What are Microservices?

The services compose together to form an application

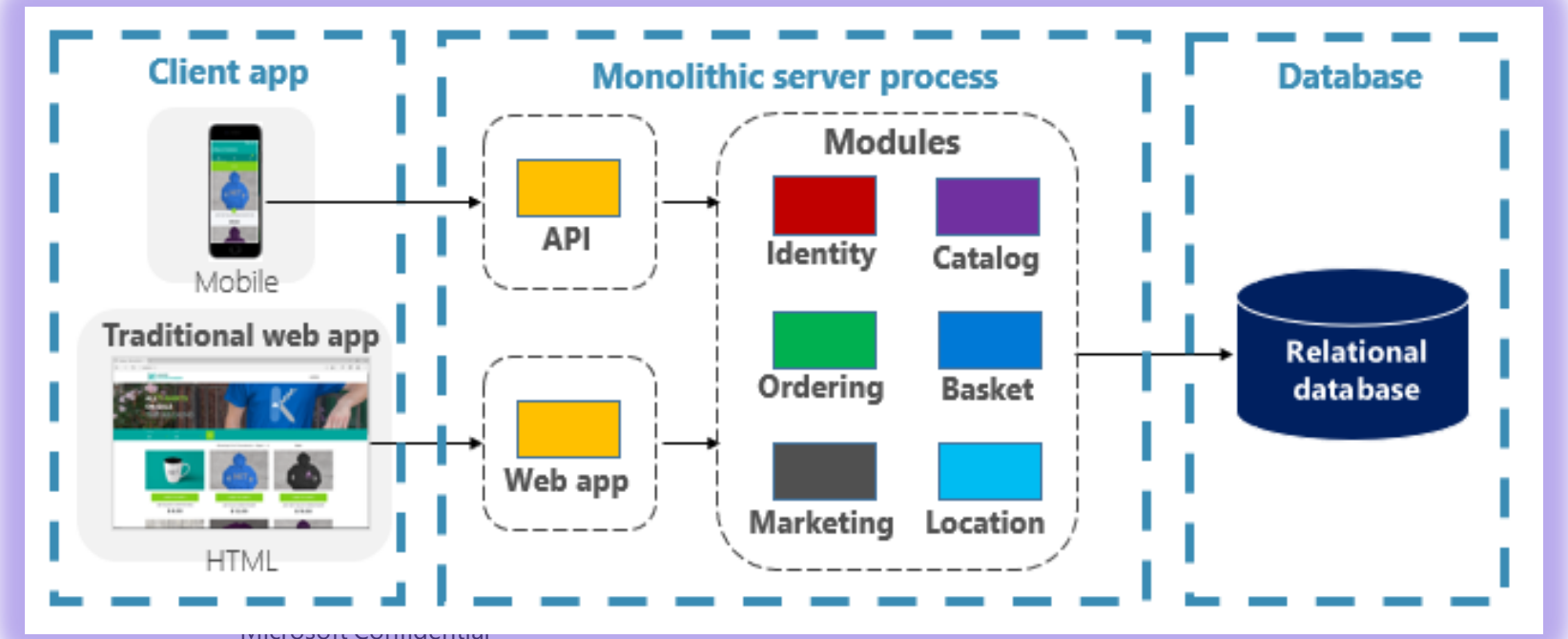
Building that New Application

- You're hired!!
- Build the next generation eCommere app – compete with Walmart and Amazon
- What is your design???

- Adhering to guidance from past 15 years
  - Create *large core* which contains…
    - Business/data access logic
    - Front-end logic
    - Shared relational database

- Build, package and deploy
- You're in business!

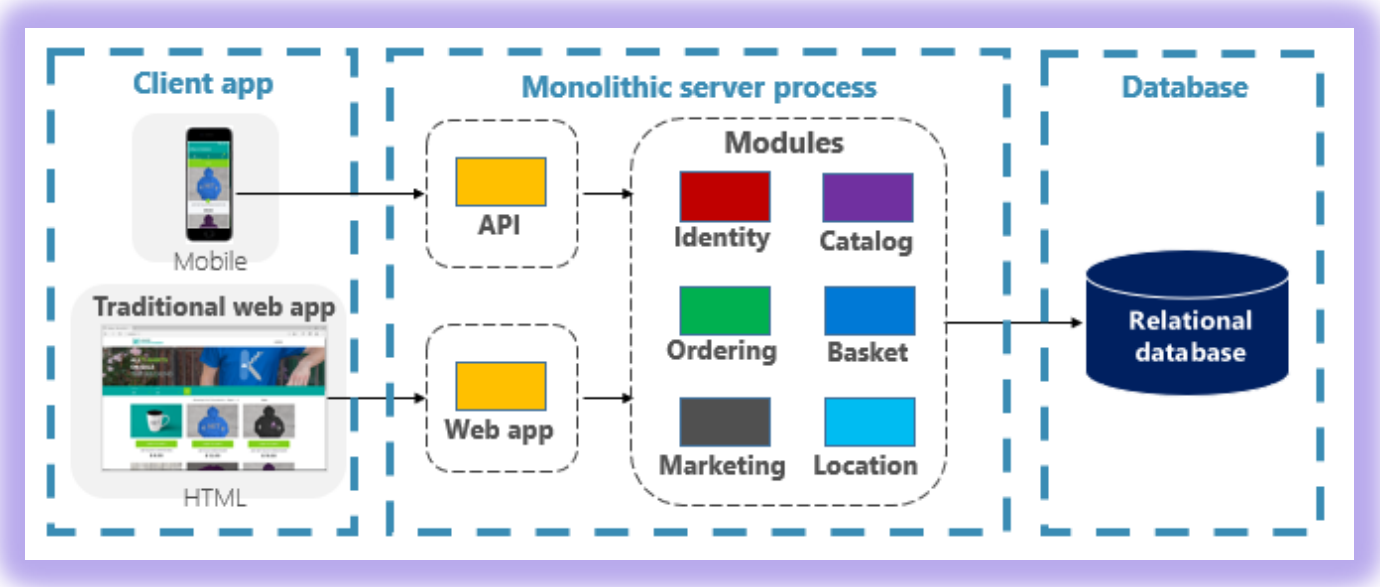
- *Takeaways…*
  - *Layered - SoC*
  - *Runs in single process*



The Monolith

- Congrats! You just built a *monolithic* app
- Not all bad…

- Can be straightforward to…
  - build
  - test
  - deploy
  - troubleshoot
  - scale



- Performance can be good (single process)
- Many successful apps are monoliths
- Your app is a hit and continues to evolve

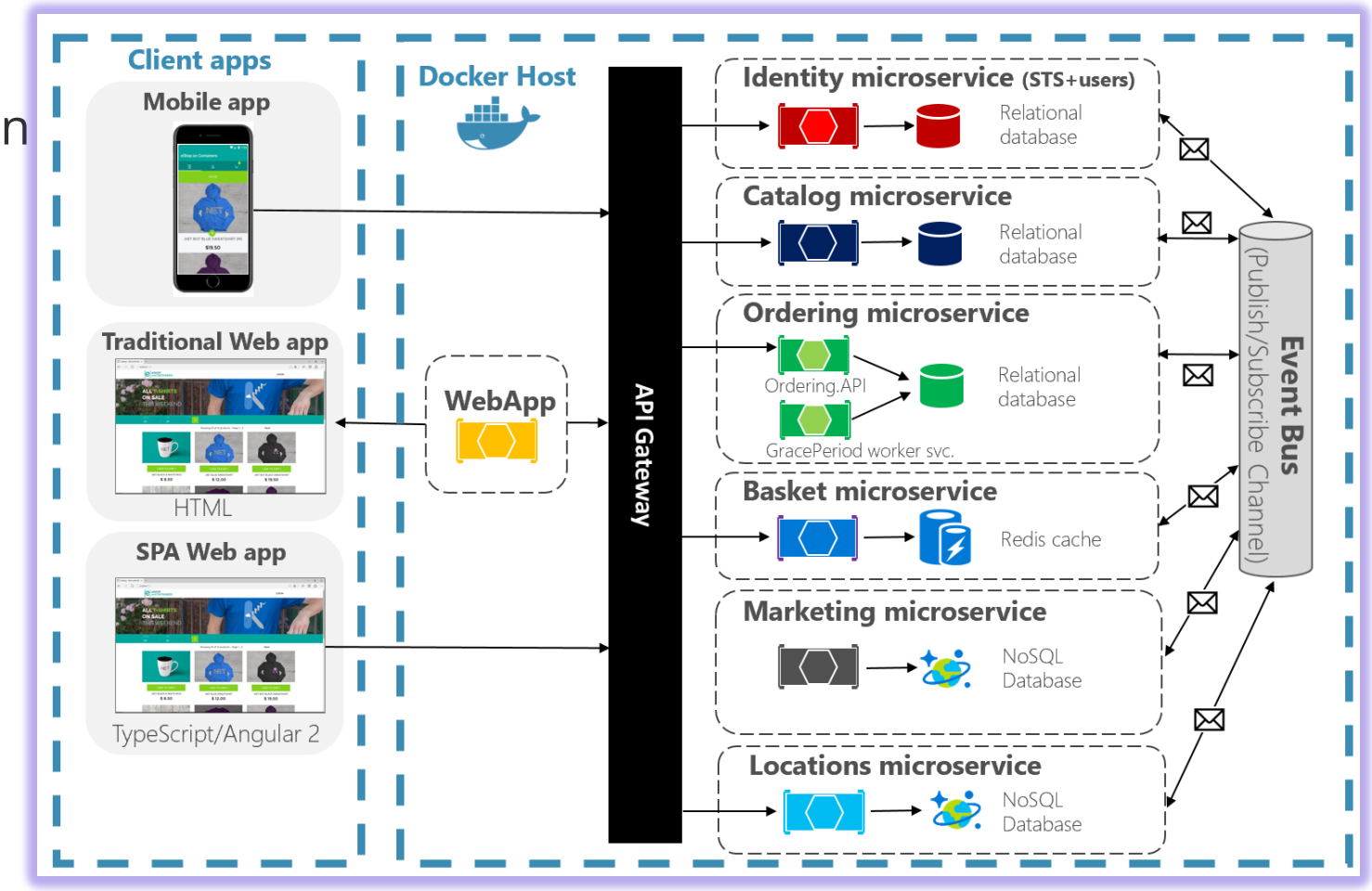
Over Time…

- You feel yourself losing control and enter *The Fear Zone*
  - The app has become overwhelmingly complicated - no single person understands it
  - You dread making changes – each causes unintended and costly *side effects*
  - Fixes/features raise anxiety - time-consuming, expensive, and error prone
  - You adamantly push back on any change/enhancement – each becomes as small as possbile
  - Instead of building innovative cool apps, your time and IT budget is sucked into support
- But there's more…
  - Each change requires a full deployment of the entire application
  - One unstable component can crash the entire application
  - Implementing new technologies and frameworks is not an option
  - Architectural erosion sets in – code base become more and more tightly coupled
  - Consultants come in and tell you to rewrite it

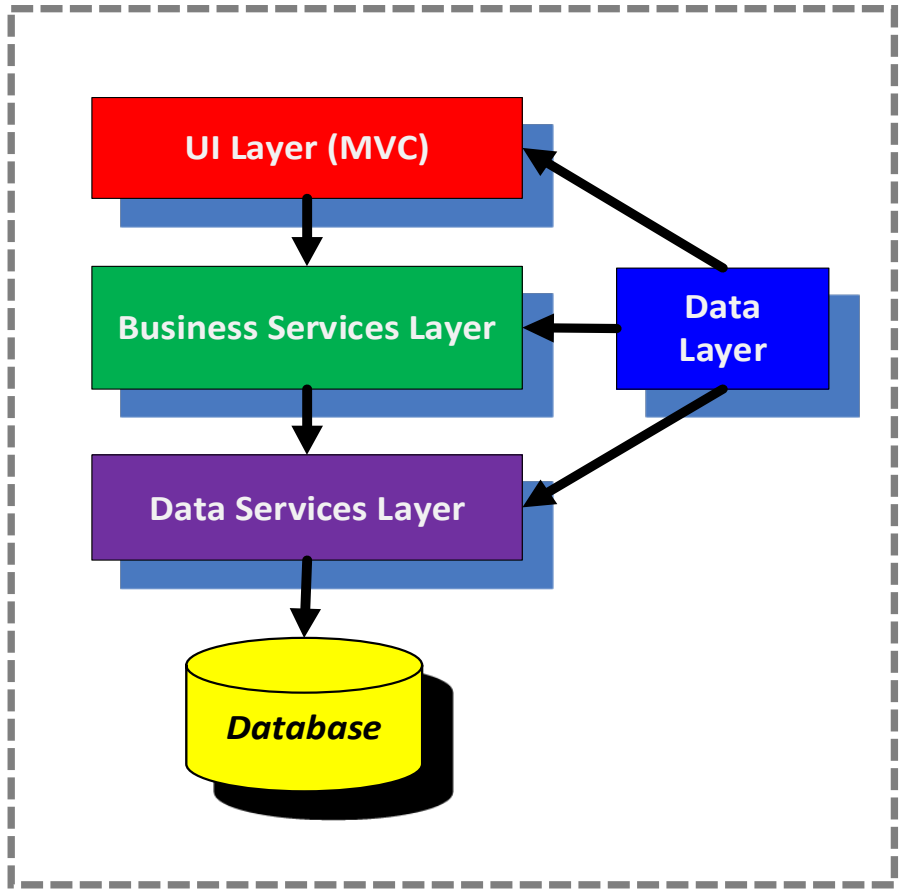
Looking Forward

- Many organizations are mitigating *Fear Zone* heart burn with microservices

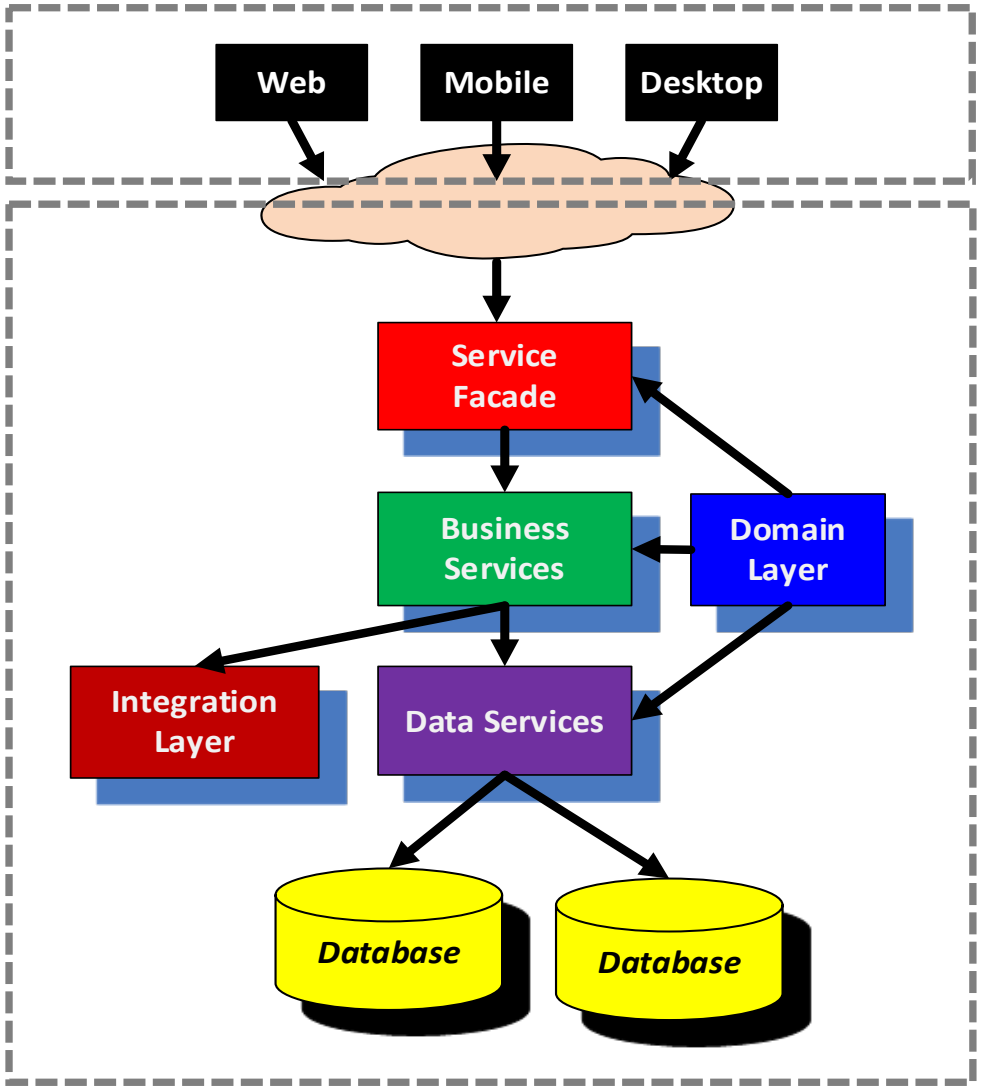
  - Small independent services
  - Platform agnostic communication
  - Distributed data
  - Asynchronous messaging



Is this a microservice app?

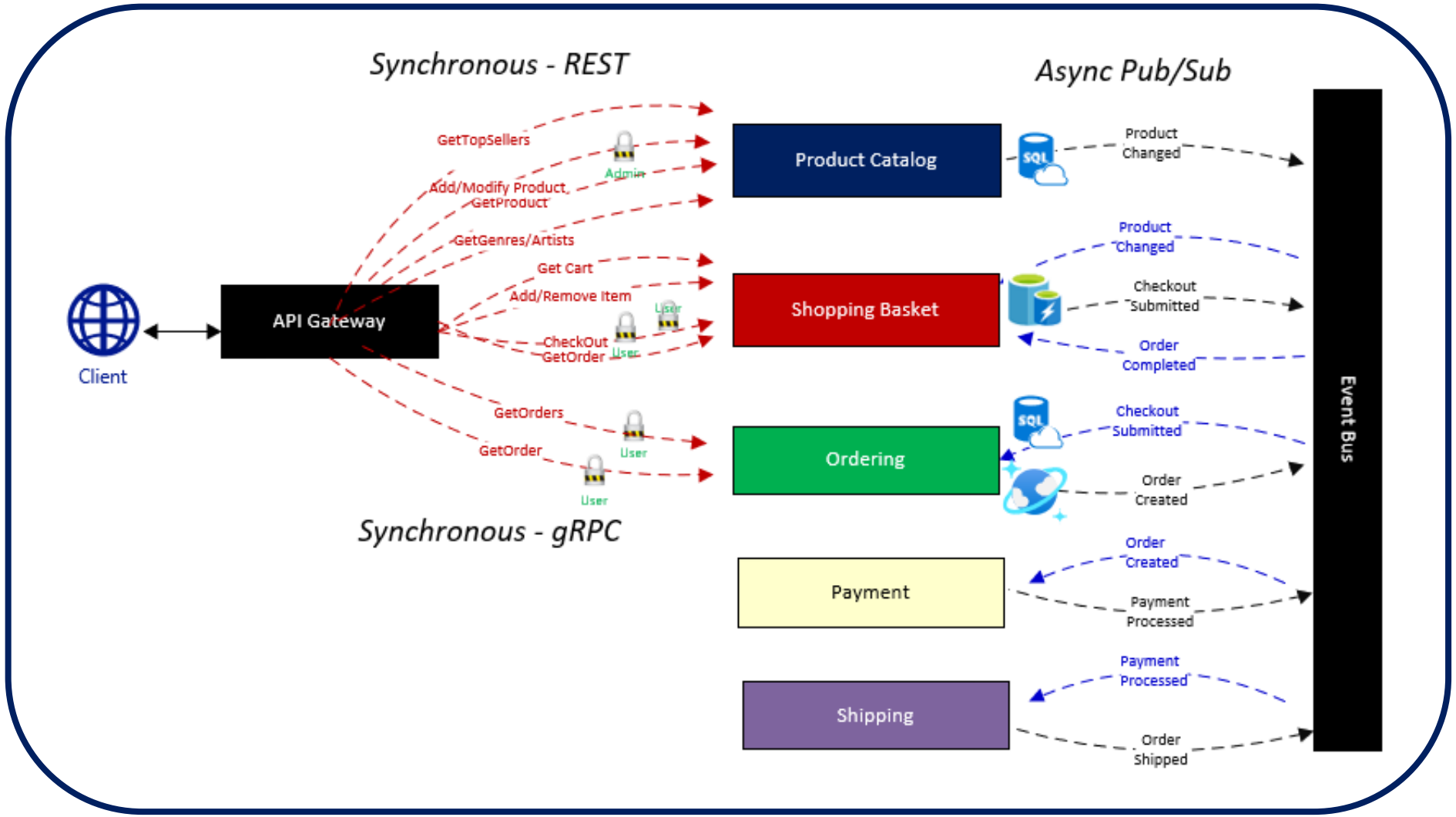


Garden Variety Web App

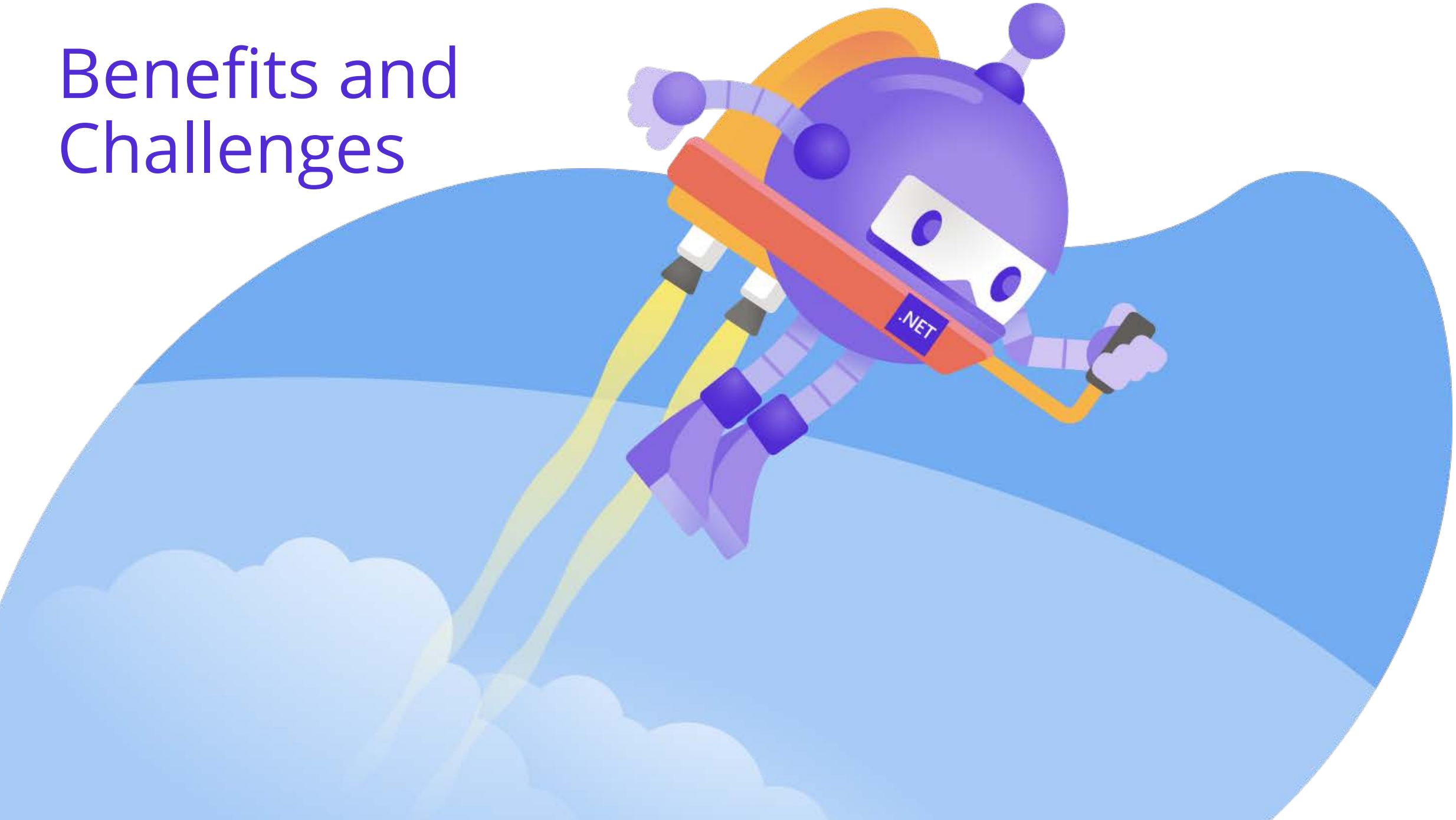


Garden Variety API App

Is this a microservice app?



Benefits and Challenges



Are these companies agile?

Numerous services deployed in production

Makes 30,000 changes per month

NETFLIX

600+ services in production

Deploy hundreds of times each day

UBER

1,000+ services (stored in 8,000 GIT repositories)

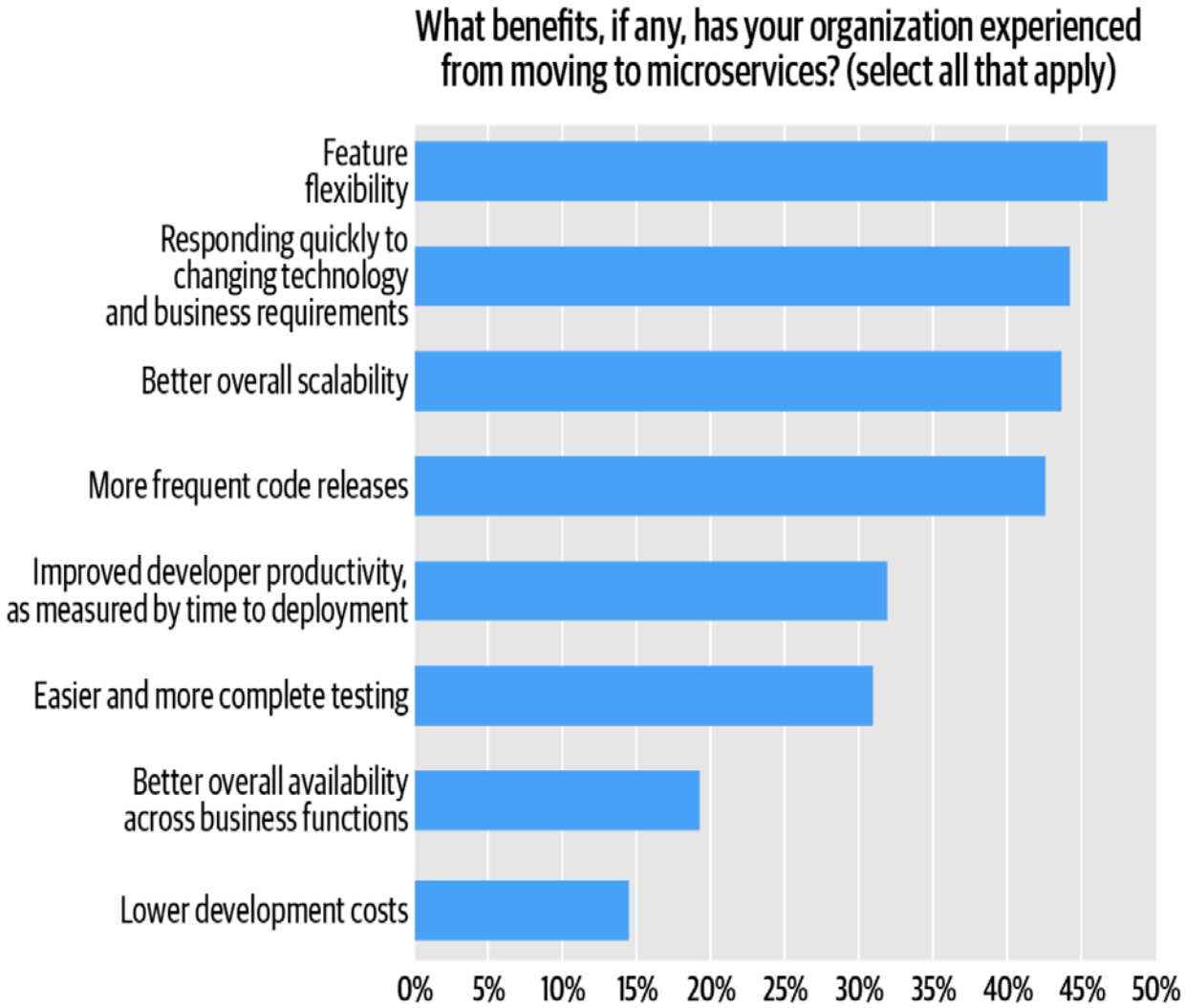
Deploys every minute during work hours

How Important is Agility?

- Perhaps the *biggest benefit* of a microservice architecture is *agility*
  - System closely aligns to business capabilities
  - Can rapidly respond to market conditions
  - Instantaneously update small areas of a live applications
  - Evolve app services independently and deploy frequently

More Benefits

- Reduced maintenance costs
    - Updates/fixes target single service, not entire application
- Fine-grained scalability
    - Scale services Independently - optimizing resource costs
- Eliminate stack lock-in
    - Optimize agility by mixing heterogenous platforms/data stores
    - Future-proof application investment against obsolete technology stacks
- Increase resiliency
    - Release with confidence - isolate failure to specific service(s)



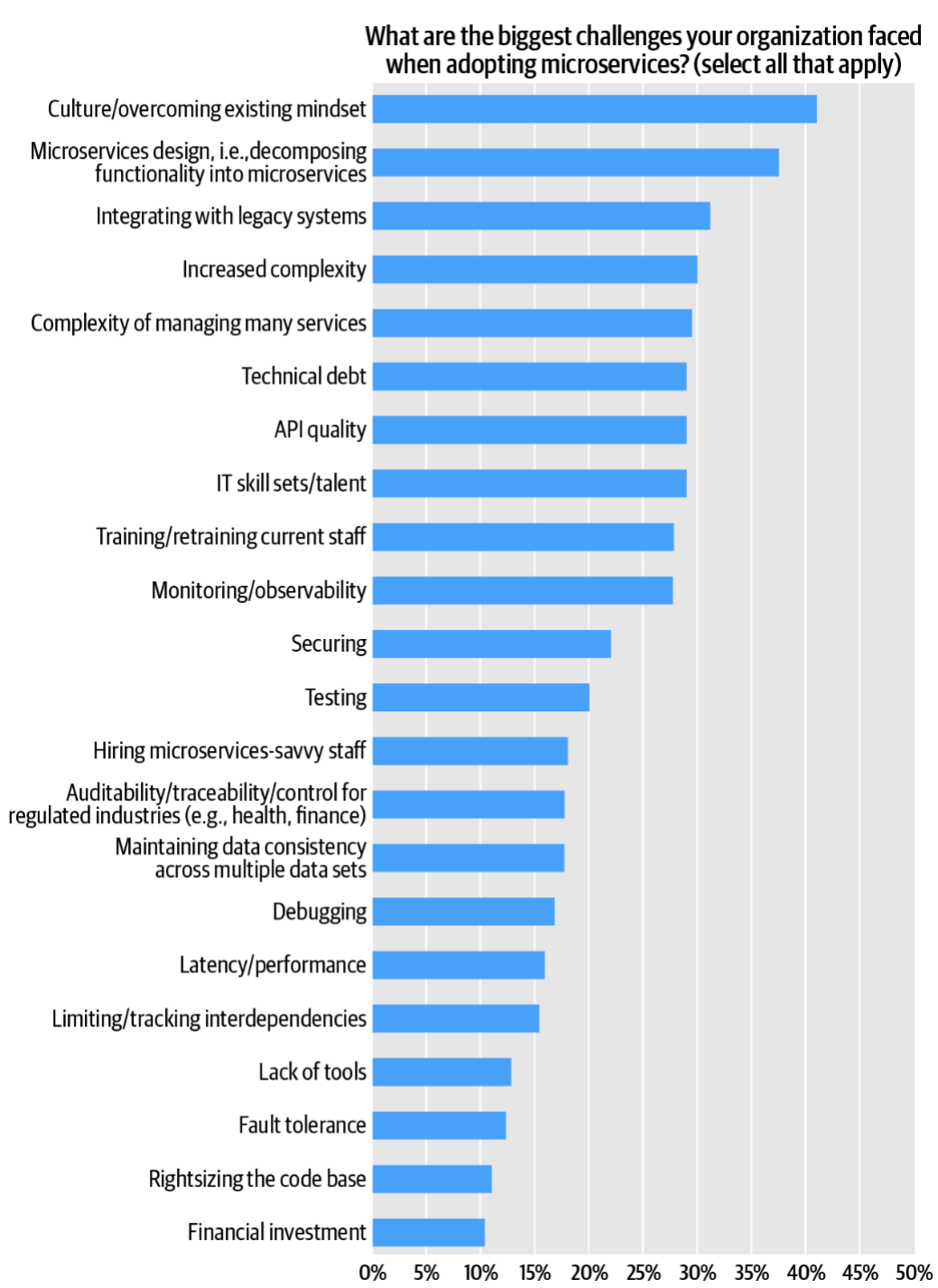
There's more…

Consider a recent O'Reilly poll of *1502 respondents*…

https://www.oreilly.com/radar/microservices-adoption-in-2020/

Microservices - Challenges

- No free lunch…
- Microservices are distributed systems and distributed systems are hard…
  - Your architectural and operational complexity (dramatically) increase
  - In-process call stack replaced with network calls, introducing network congestion, latency, partitions, and transient faults
  - Data becomes distributed - move from immediate to eventual consistency
  - Versioning gets complicated
  - Service discovery and routing concerns become complex
  - Automation, orchestration, and monitoring become mandatory



There's more...

https://www.oreilly.com/radar/microservices-adoption-in-2020/

Microservice Candidates

- Consider microservices for…
  - Complex applications that are difficult to manage and maintain
  - Strategic systems that need to align business capabilities/features
  - Systems that require a high-release velocity
    - Immediate feature releases with high confidence
    - Instantaneously update small areas of live application without downtime
  - Systems that require heterogenous technology
  - Applications with components that must scale independently

Summary

- Microservices is an architecture for decoupling large, monolithic applications into a small, independent services
- While the architecture raises many challenges, it offers several benefits:
- The agility of independent service deployment is the key
- .NET Core and Azure offer a tremendous amount of tooling and resources to help you construct and deploy microservice applications