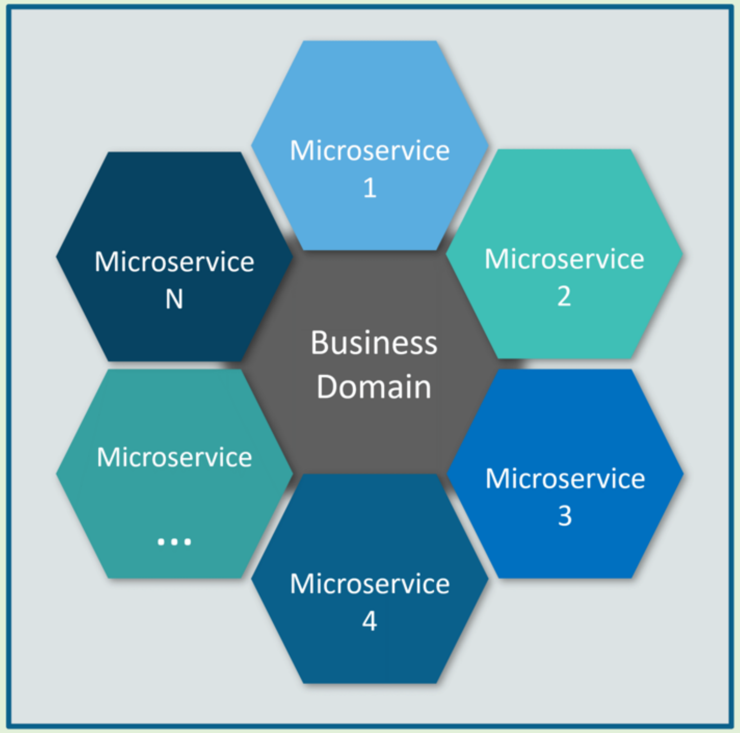


Microservices Architecture

Vidya Vrat Agarwal

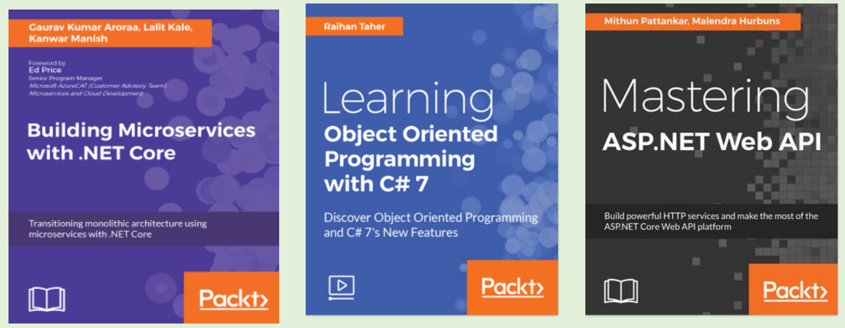
www.MyPassionFor.Net | @dotNetAuthor

https://www.linkedin.com/in/vidyavrat/

https://github.com/vidyavrat

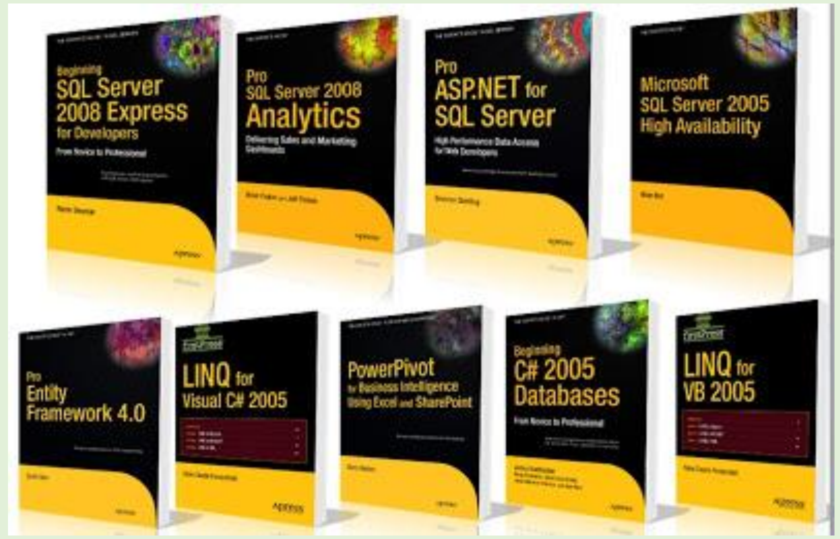
About Me

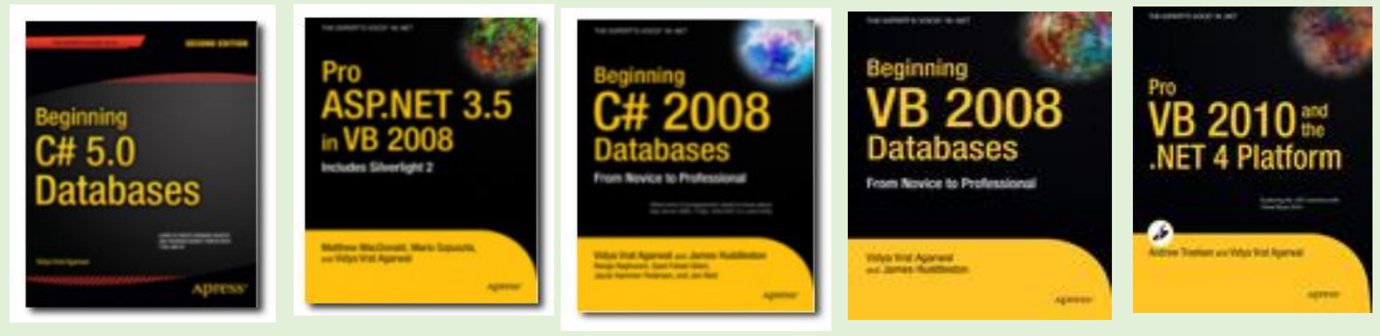
- 18+ years of industry experience
- Principal Architect with T-Mobile
- Microsoft MVP | C# Corner MVP
- TOGAF Certified Architect
- Certified Scrum Master (CSM)
- Microsoft Certified (MCT, MCSD / MCAD .NET, MCTS etc.)
- Published Author (5), and Technical Reviewer (over a dozen)


Gaurav Kumar Aroraa, Lalit Kale, Kanwar Manish
Building Microservices with .NET Core
Transitioning monolithic architecture using microservices with .NET Core
Packt

Raihan Taher
Learning Object Oriented Programming with C# 7
Discover Object Oriented Programming and C# 7's New Features
Packt

Mithun Pattankar, Malendra Hurbuns
Mastering ASP.NET Web API
Build powerful HTTP services and make the most of the ASP.NET Core Web API platform
Packt


Beginning SQL Server 2008 Express for Developers
Pro SQL Server 2008 Analytics
Pro ASP.NET for SQL Server
Microsoft SQL Server 2005 High Availability
Pro Entity Framework 4.0
LINQ for Visual C# 2005
PowerPivot for Business Intelligence Using Excel and SharePoint
Beginning C# 2005 Databases
LINQ for VB 2005

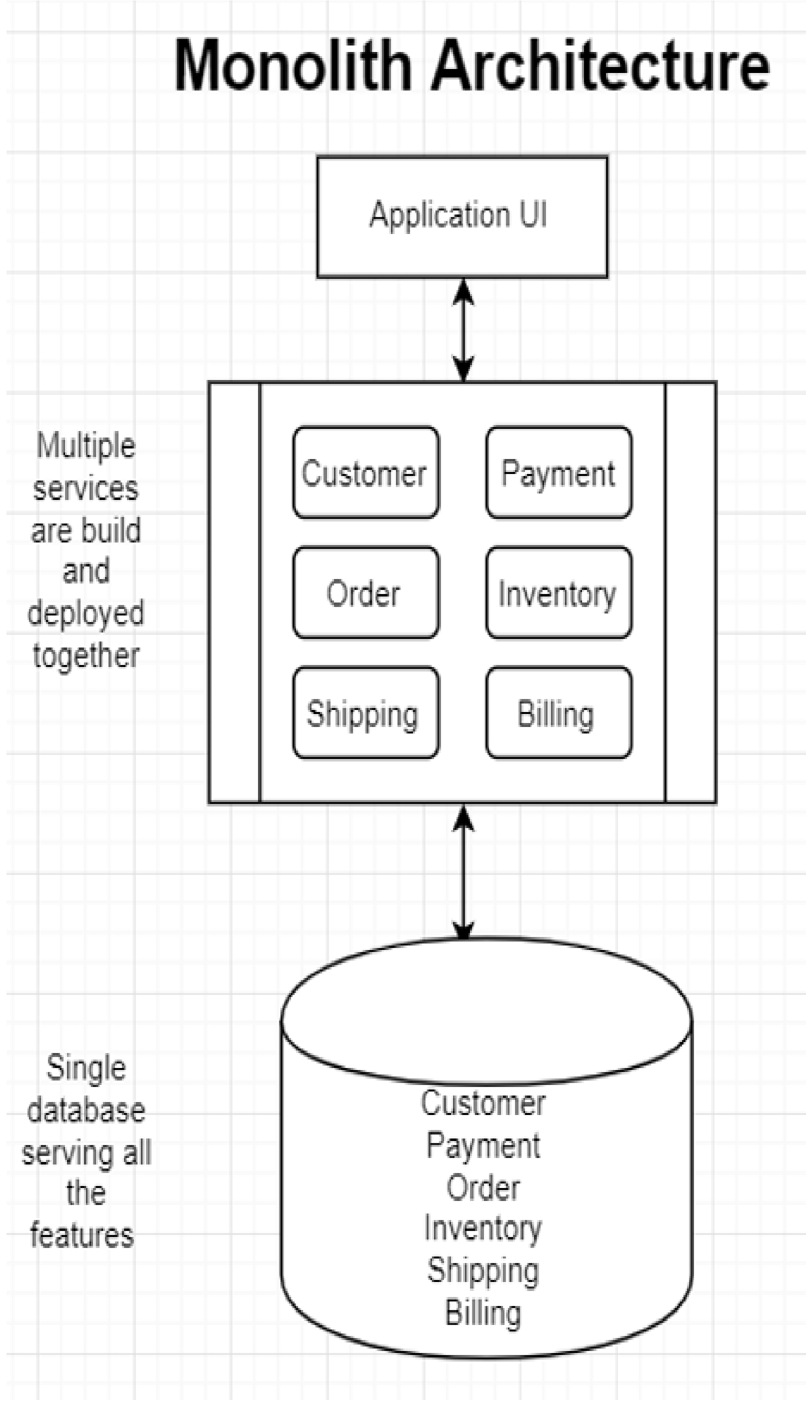

Beginning C# 5.0 Databases
Pro ASP.NET 3.5 in VB 2008
Beginning C# 2008 Databases
Beginning VB 2008 Databases
Pro VB 2010 and the .NET 4 Platform

Hands-On Parallel Programming with C# 8 and .NET Core 3
Packt
Integrating Serverless Architecture
Apress
Developing Applications with Azure Active Directory
Principles of Authentication and Authorization for Architects and Developers
Manas Mayank
Mohit Garg
Apress

MVP Microsoft Most Valuable Professional
C# Corner MVP
The Open Group TOGAF® 9 Certified
Scrum Alliance®

Monolith (SOA)

- Think of any MVC pattern-based API code base, where all your controllers and POJOs (Plain Old Java Objects) or POCOs (Plain Old C# Objects) were developed, build and deployed as a single unit, and for almost all the times a single data store was used for the enterprise.



Monolith Pros and Cons:

Pros:

•**Fewer Cross-cutting Concerns**: The major advantage of the monolithic architecture is that most apps typically have a large number of cross-cutting concerns, such as logging, rate limiting, and security features such audit trails and DOS protection. When everything is running through the same app, it's easy to hook up components to those cross-cutting concerns.

•**Less Operational Overhead**: Having one [large] application means there's only one application you need to set up logging, monitoring, testing for. It's also generally less complex to deploy.

•**Performance**: There can also be performance advantages due to shared-memory access is faster than inter-process communication (IPC) or Remote Procedure Calls (RPC).

Cons:

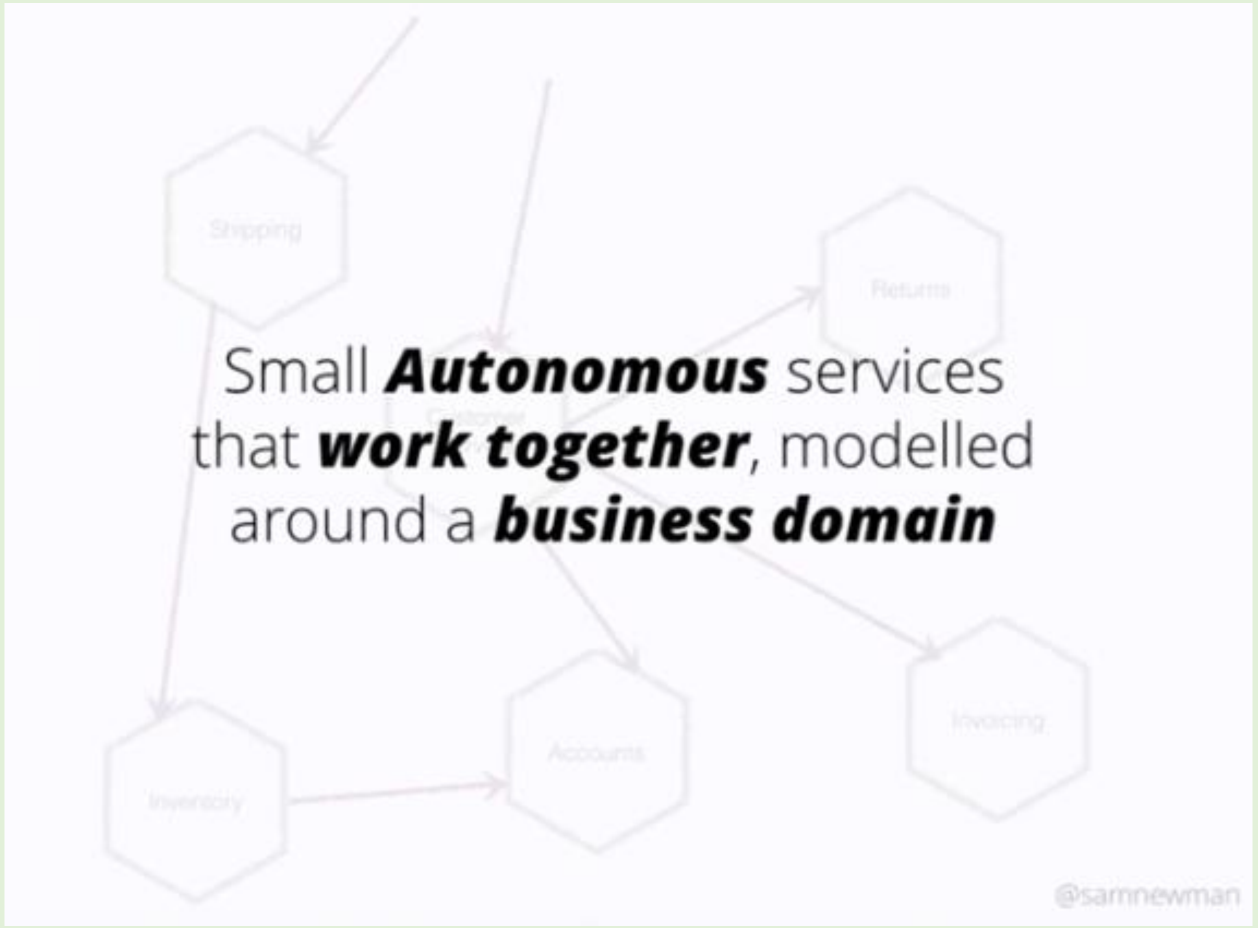
•**Deploy all or none:** When a new change needs to be pushed, whole service needs to be deployed.

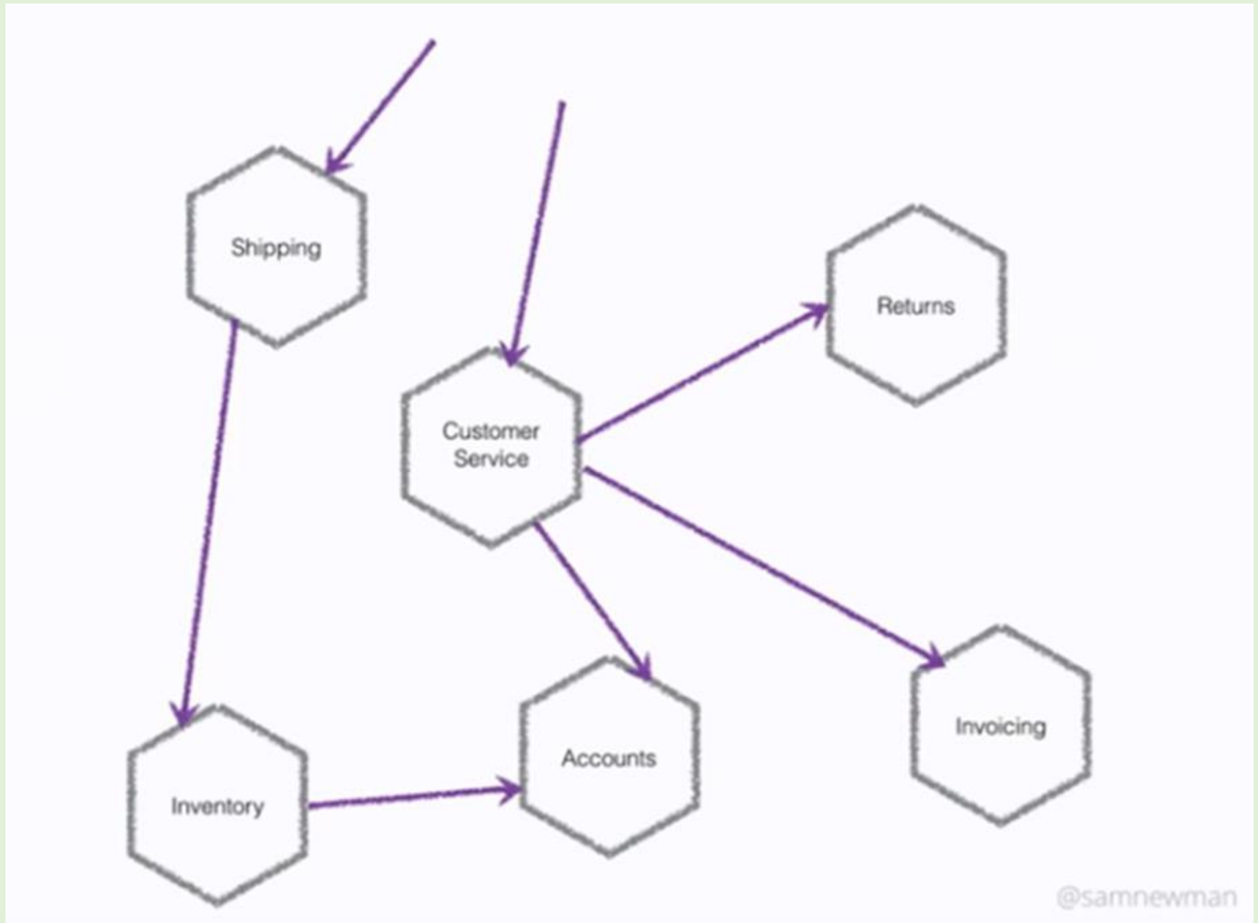
•**Scale all or none:** Scale up/down, its for entire functionality.

•**Single point of failure:** if a server is down, entire functionality is broken.

What is a Microservice?

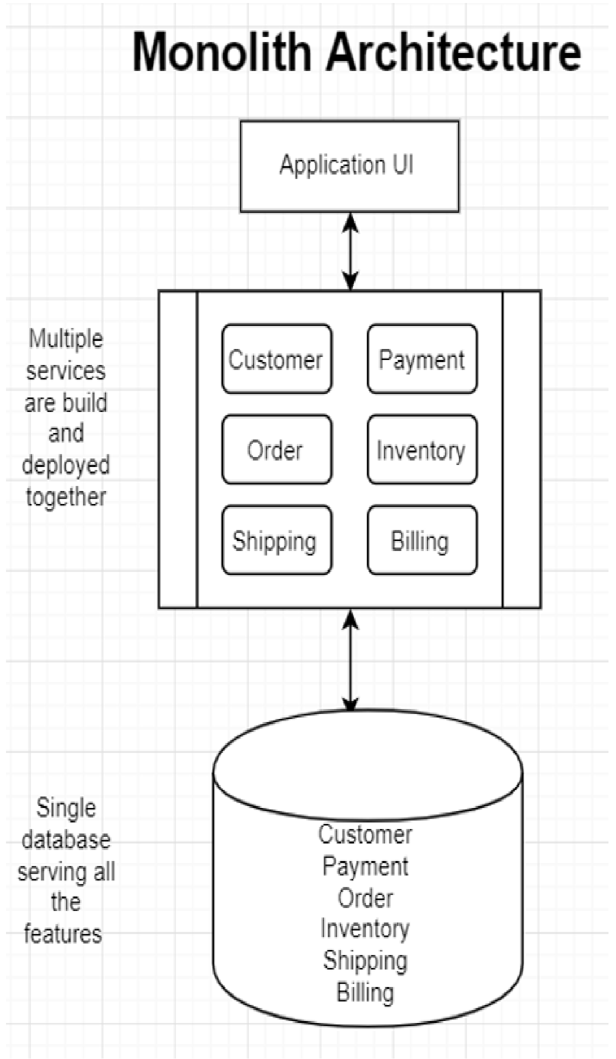
- "Microservice as a tightly scoped, loosely coupled, strongly encapsulated, independently deployable, and independently scalable application component."
- Global Microservice Architecture Market anticipated accreting to US$ 33 Billion by 2023.

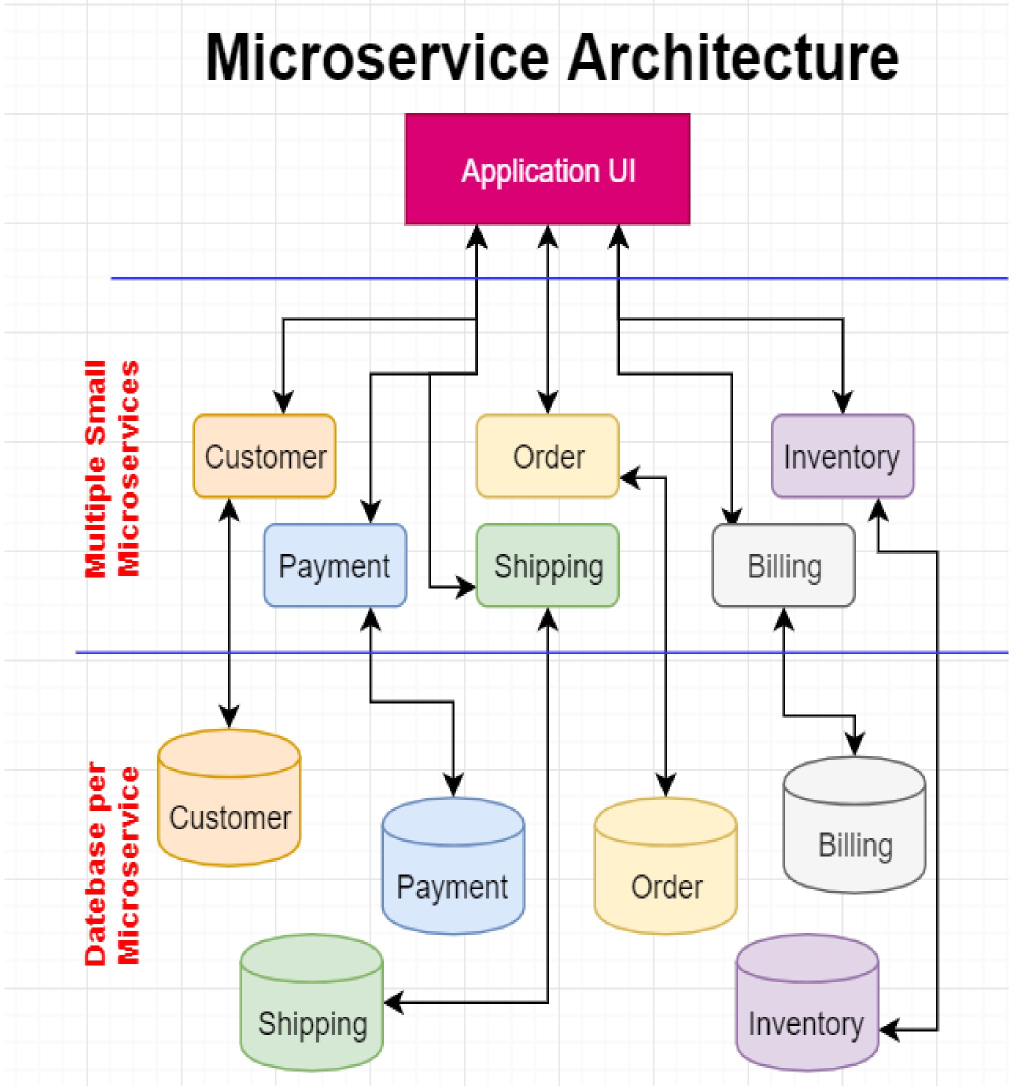

Small **Autonomous** services that **work together**, modelled around a **business domain**
@samnewman



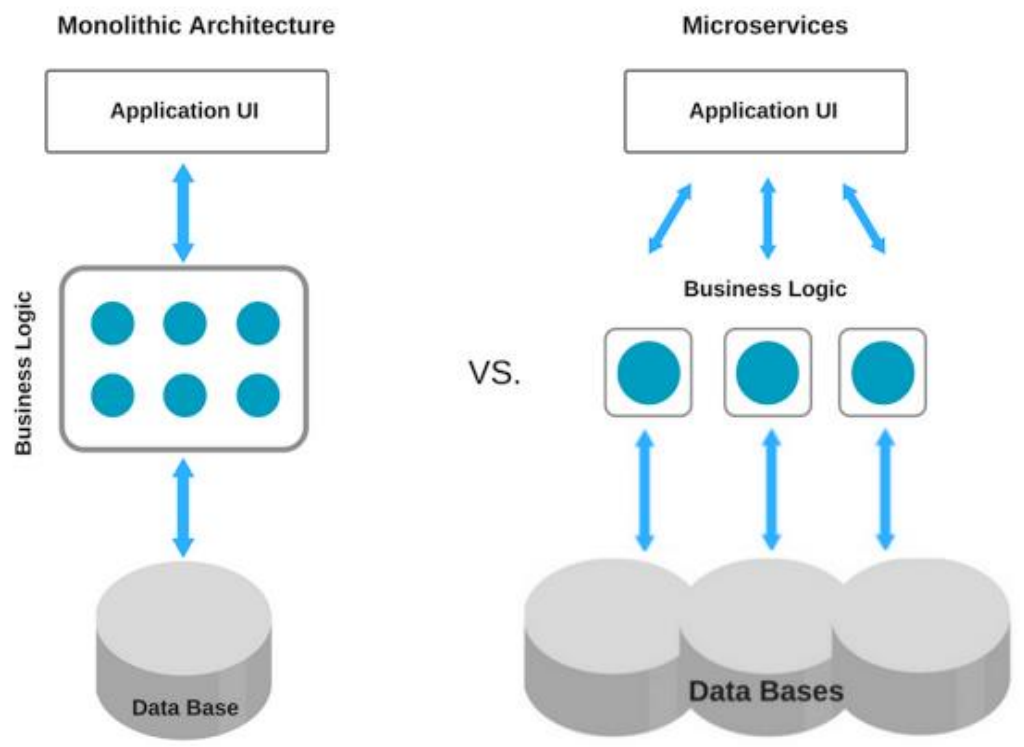
Finding a Bounded Context

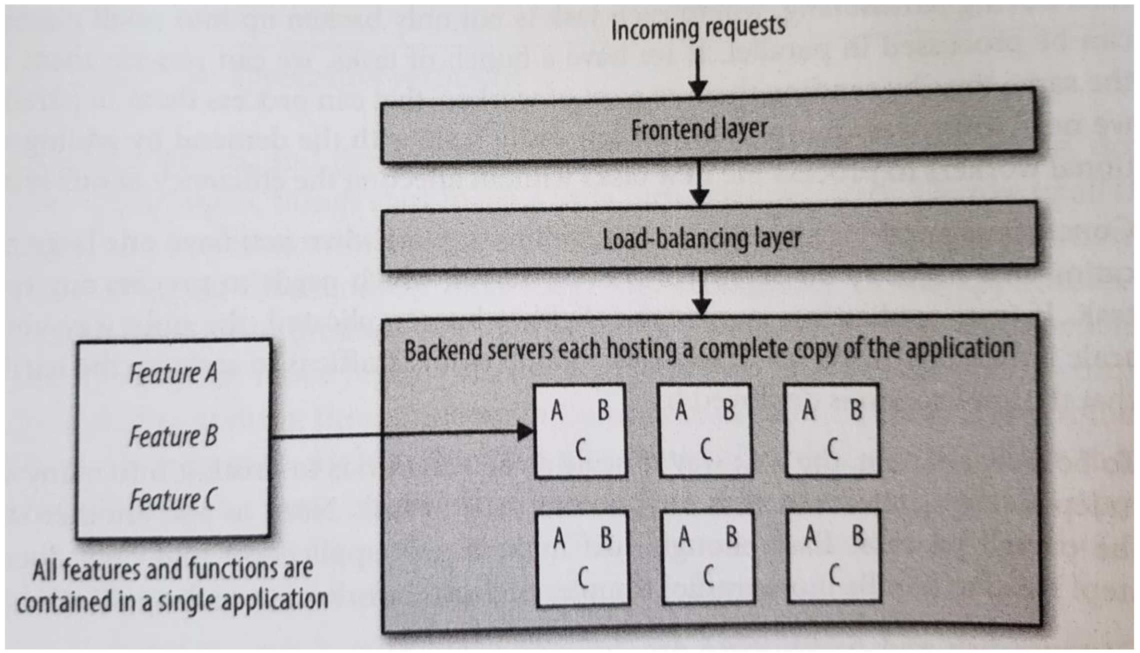
A bounded context is an explicit boundary within which a domain model exists.



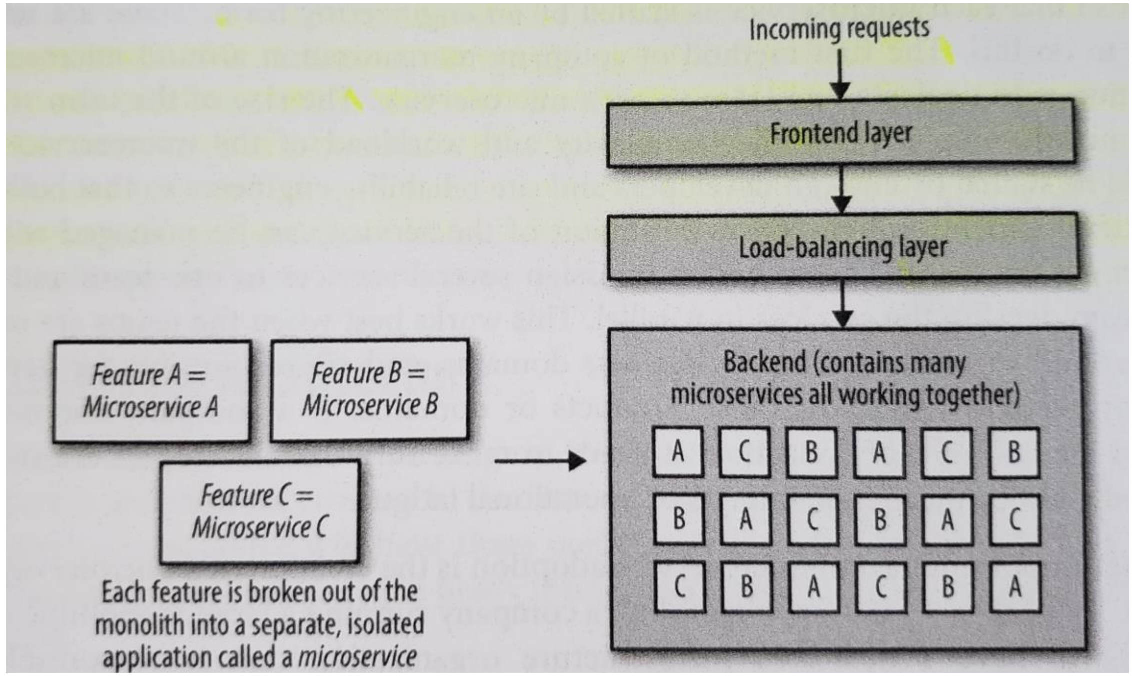


The difference between the monolithic and microservices architecture



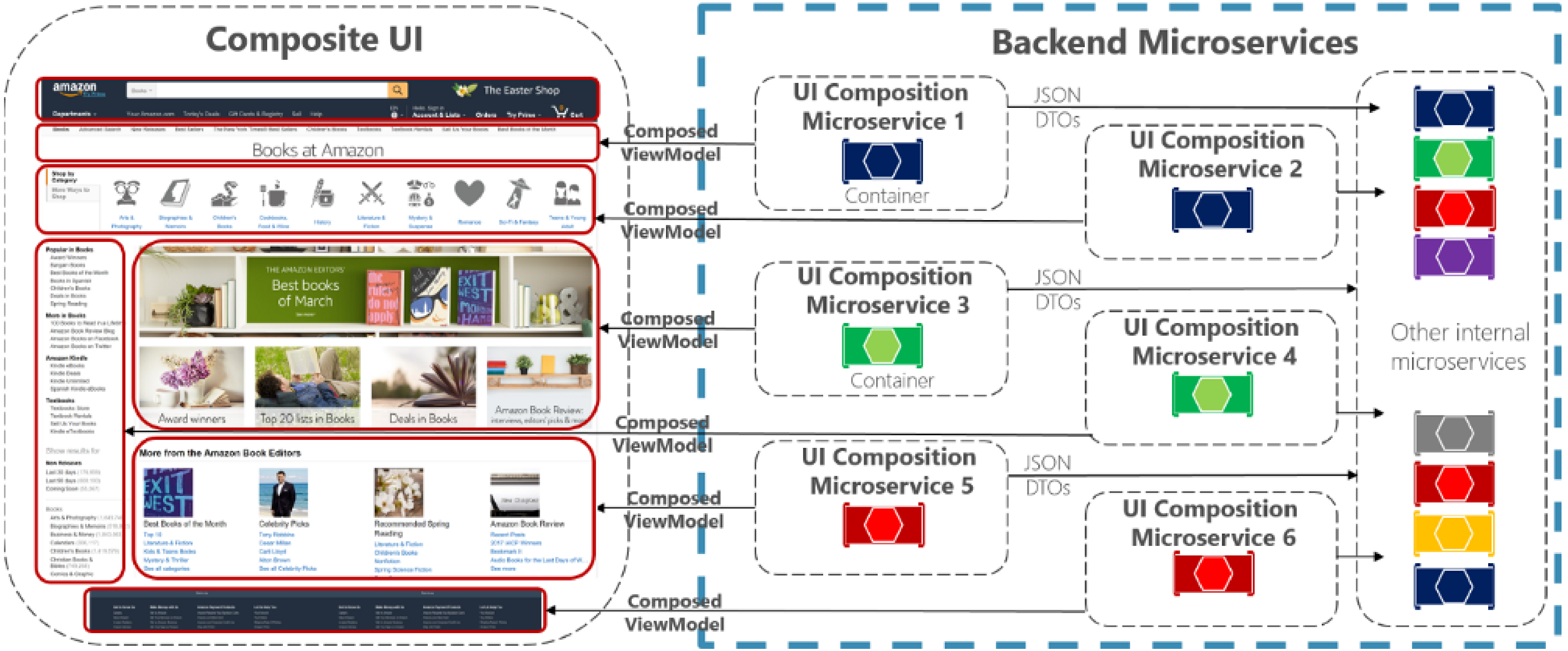


Incoming requests
Frontend layer
Load-balancing layer
Backend servers each hosting a complete copy of the application
Feature A
Feature B
Feature C
A B C
All features and functions are contained in a single application



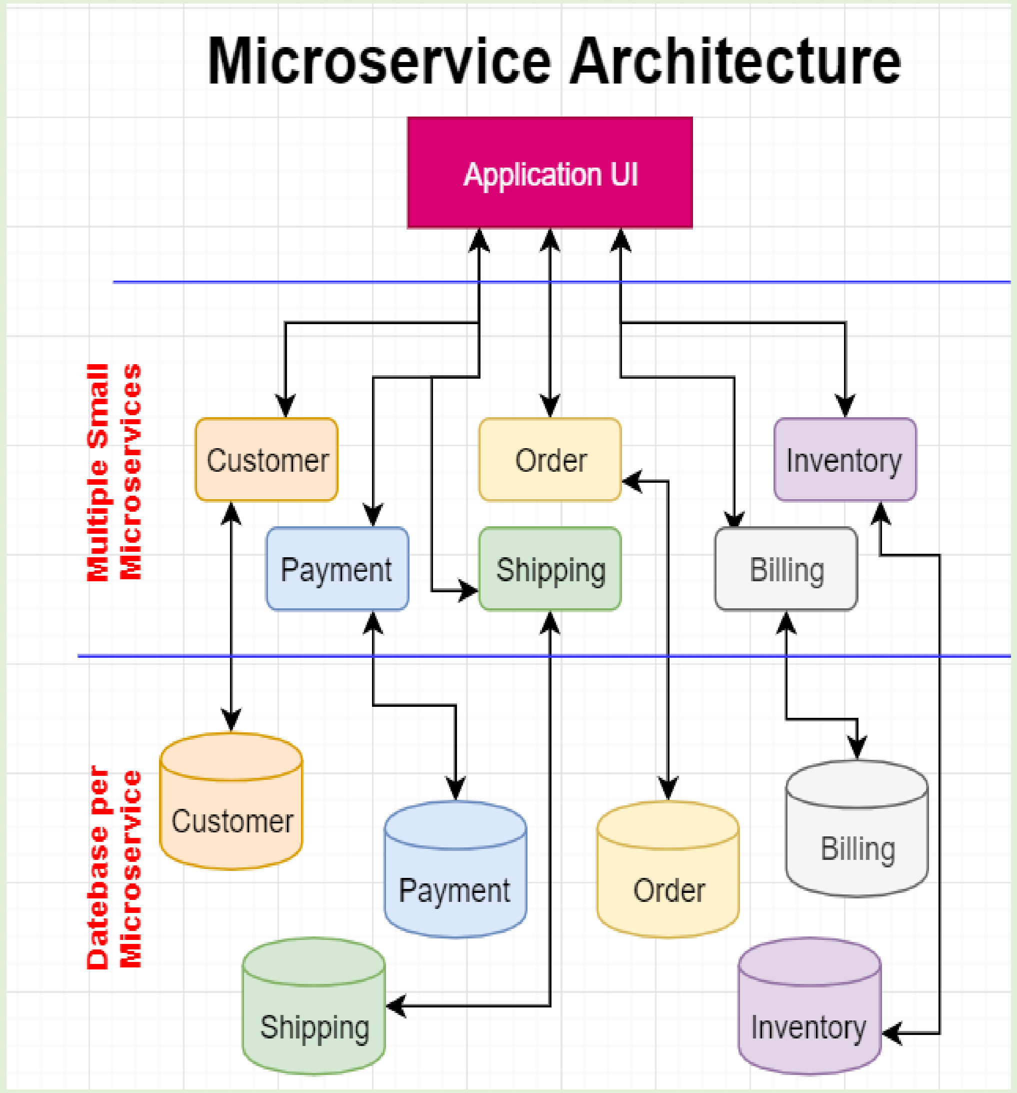
Incoming requests
Frontend layer
Load-balancing layer
Backend (contains many microservices all working together)
Feature A = Microservice A
Feature B = Microservice B
Feature C = Microservice C
A C B A C B
B A C B A C
C B A C B A
Each feature is broken out of the monolith into a separate, isolated application called a microservice

Microservices In Action



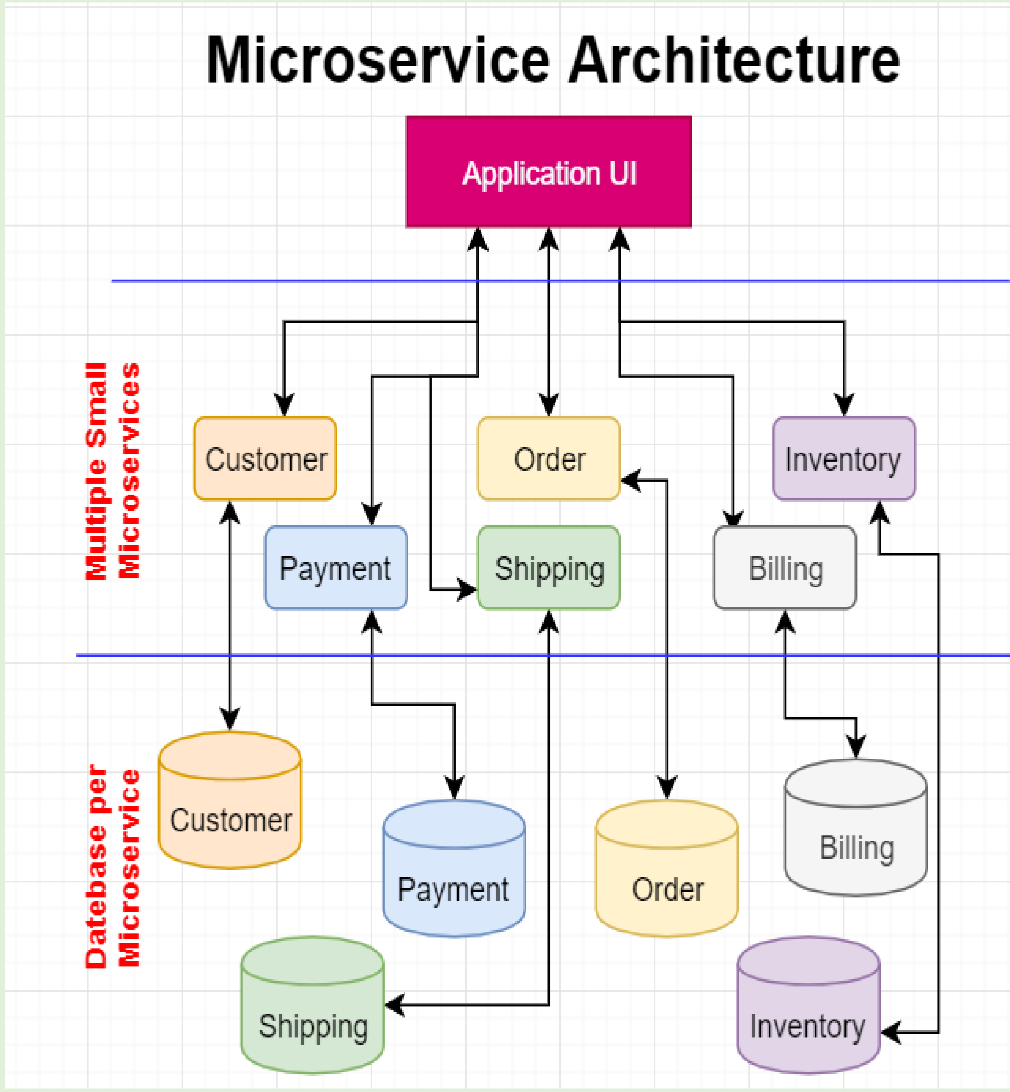
Advantages of Microservice

- Scalability
- Easier Deployments
- Problem isolation
- Single Responsibility
- Deep domain knowledge
- Polyglot programming



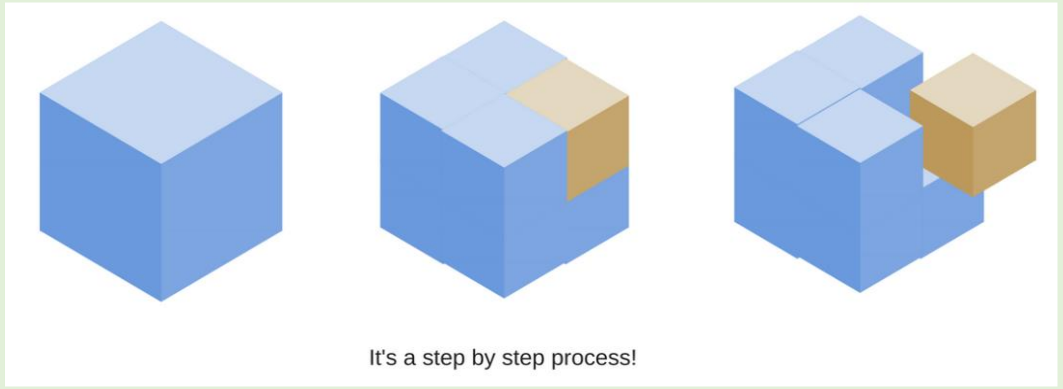
Disadvantages of Microservice

- Cultural Change
- More Expensive
- Complexity
- Less Productivity
- Communication between services
- Harder to do integration tests
- Well thought architecture right from beginning
- Complexity

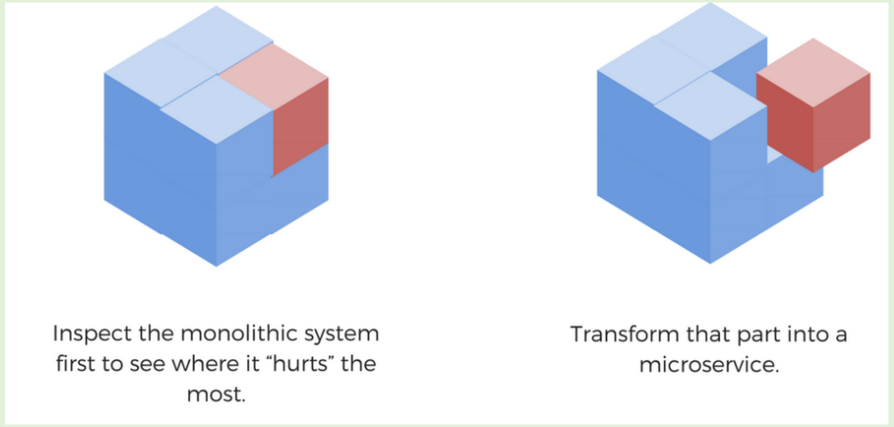


How to Migrate from Monolith to Microservice Architecture ?

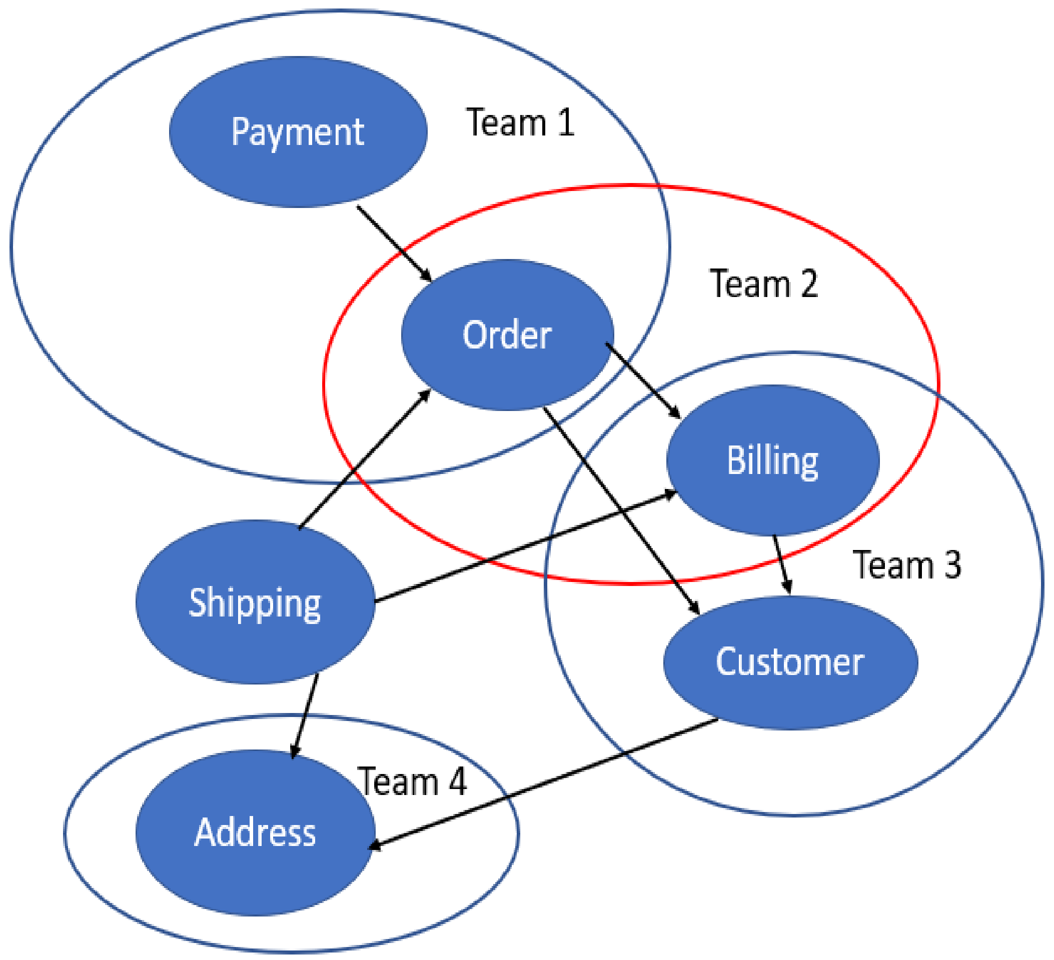
- Don’t switch from monolith to microservice all at once.



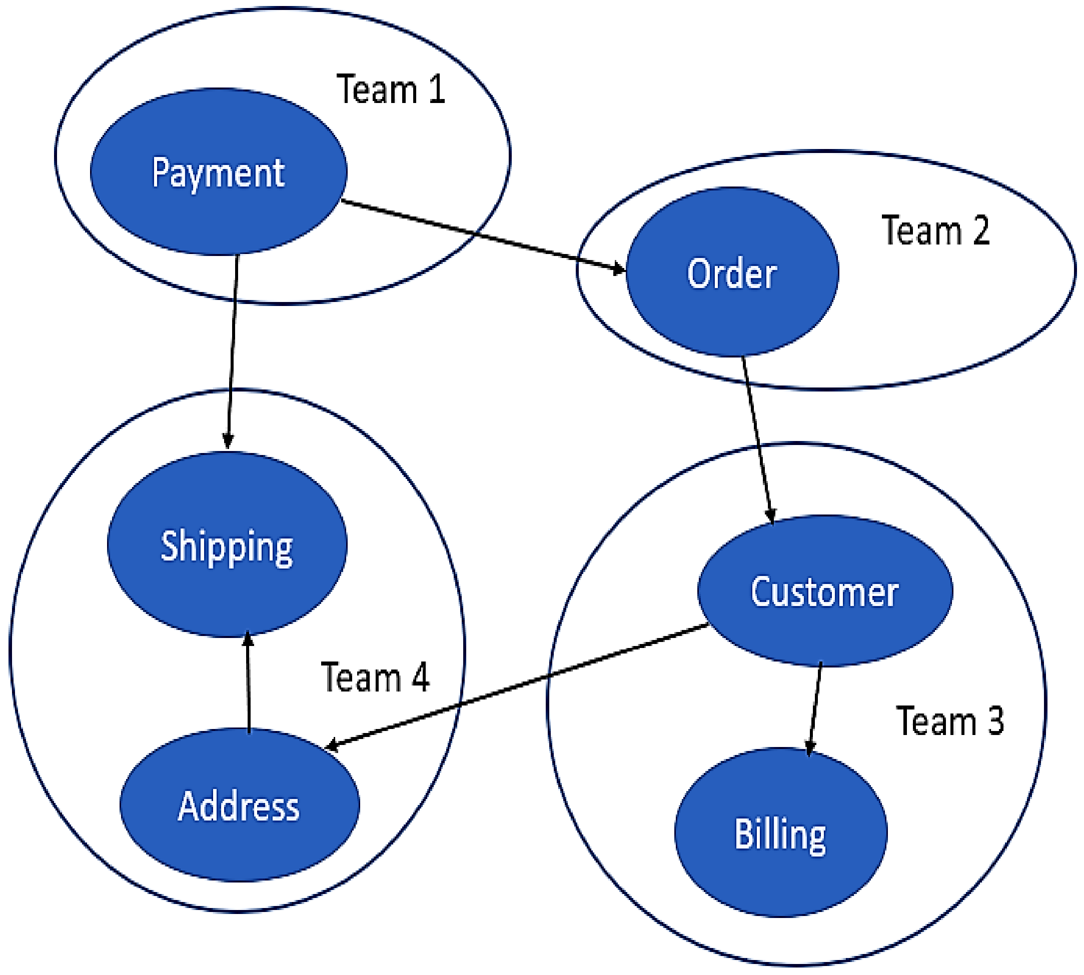
- Divide and conquer


Inspect the monolithic system first to see where it “hurts” the most.
Transform that part into a microservice.

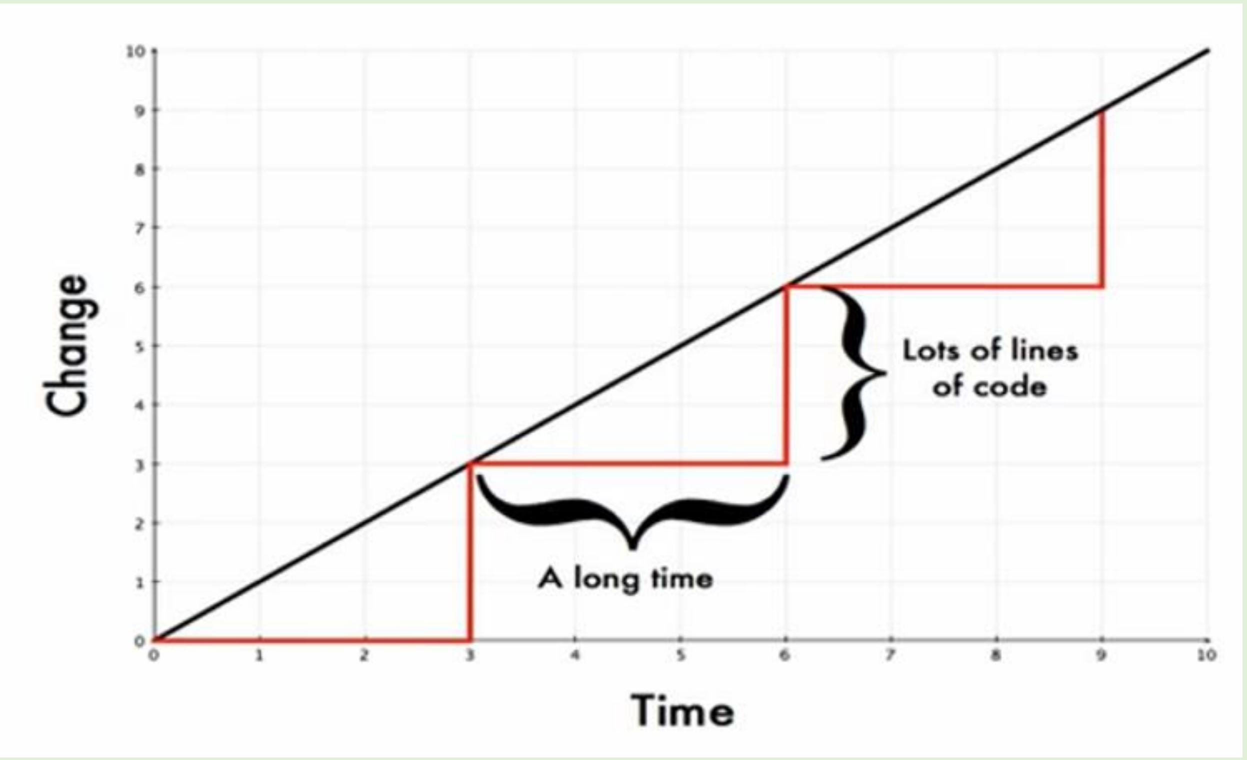
Service Ownership

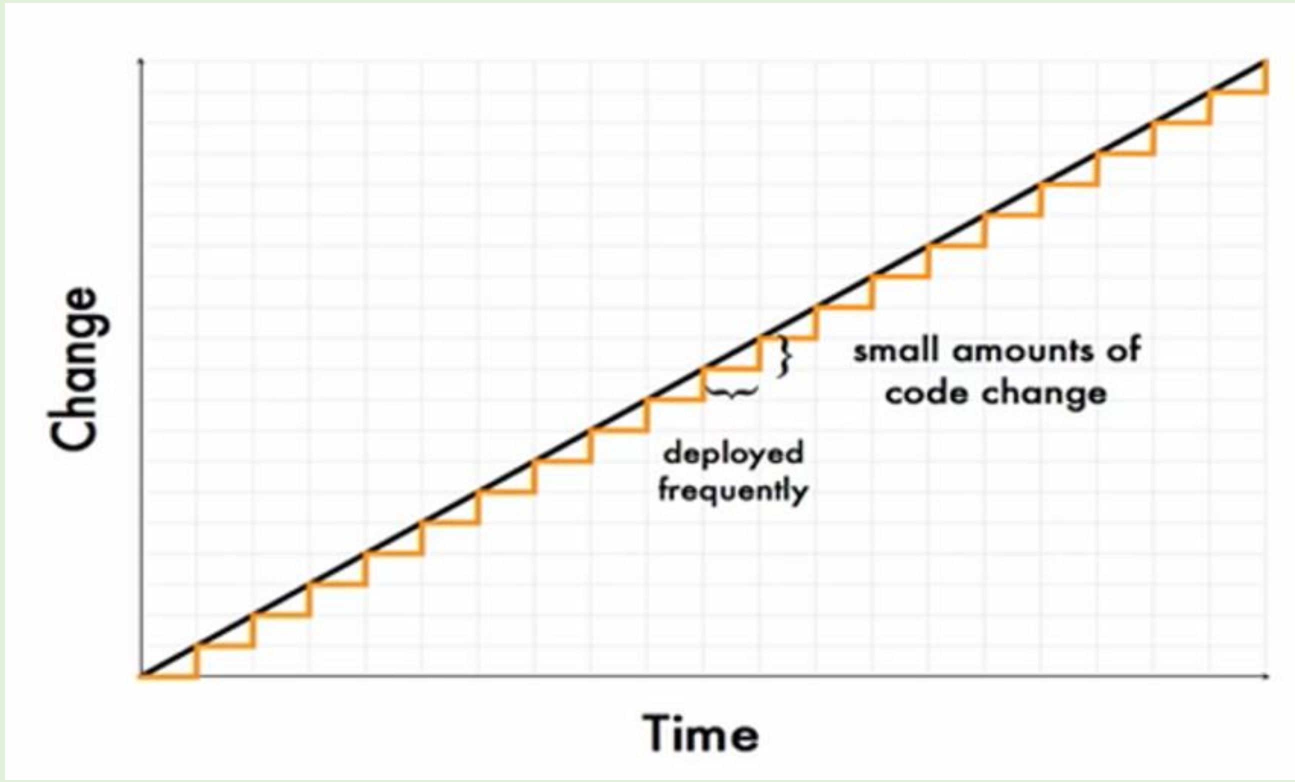


STOSA (Single Team Owned Service Architecture)



Delivery Cycles – Monolith vs Microservice

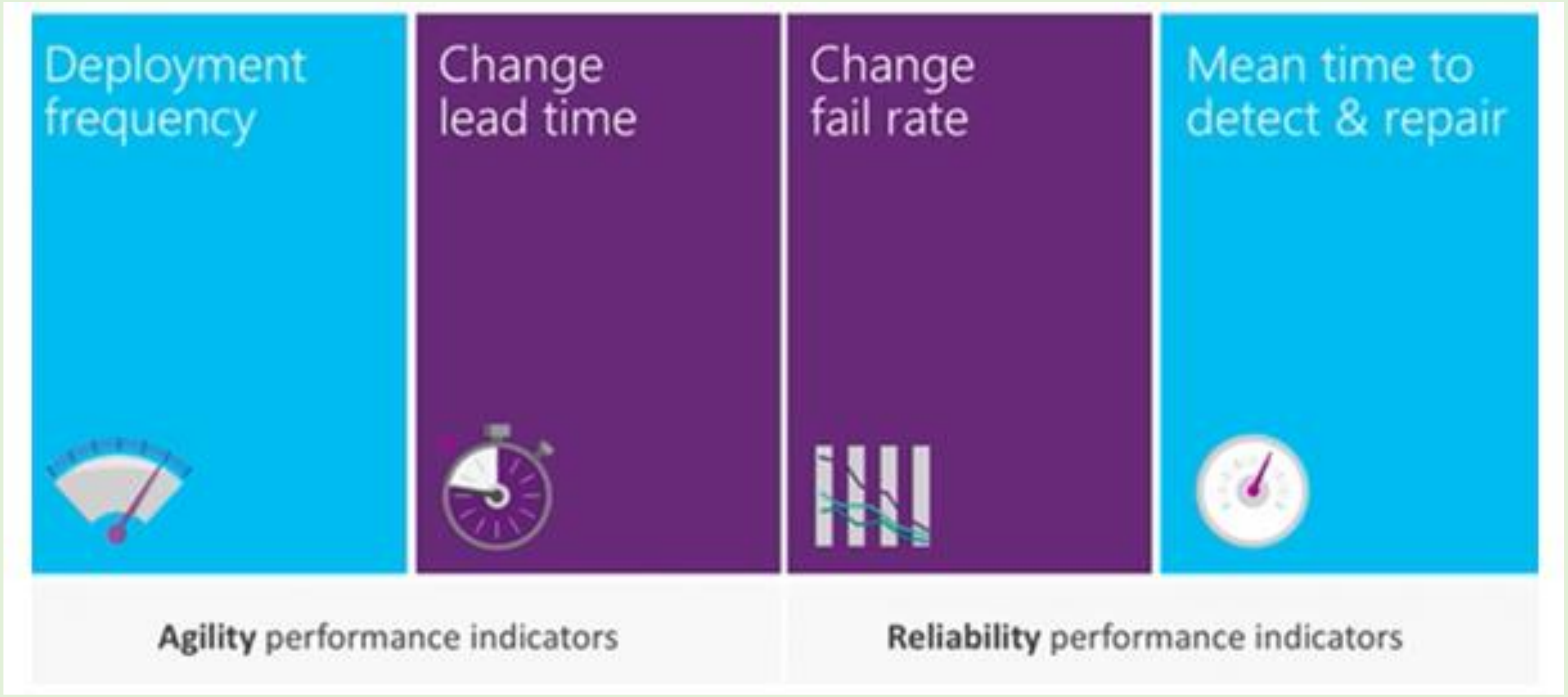




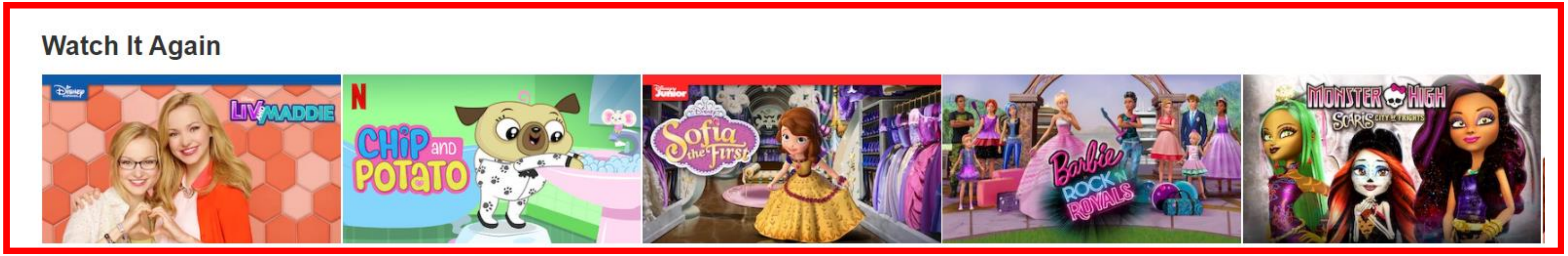
DevOps Practices a must for Microservice

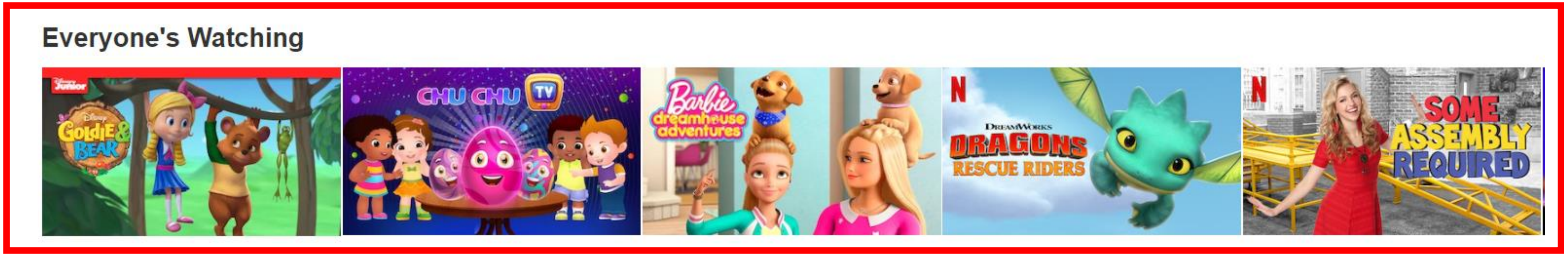
- Configuration Management
- Release Management
- Continuous Integration
- Continuous Deployment
- Infrastructure as Code
- Test Automation
- Application Performance Monitoring

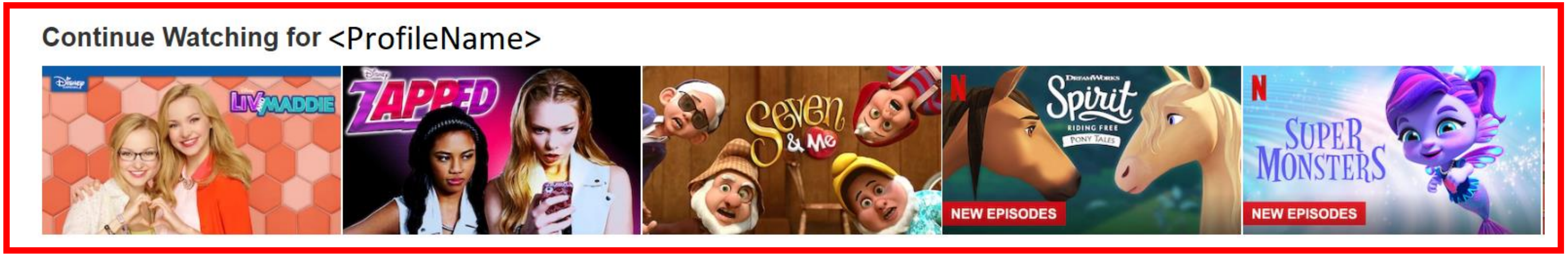
DevOps Metrics with Microservice


Deployment frequency
Change lead time
Change fail rate
Mean time to detect & repair
Agility performance indicators
Reliability performance indicators

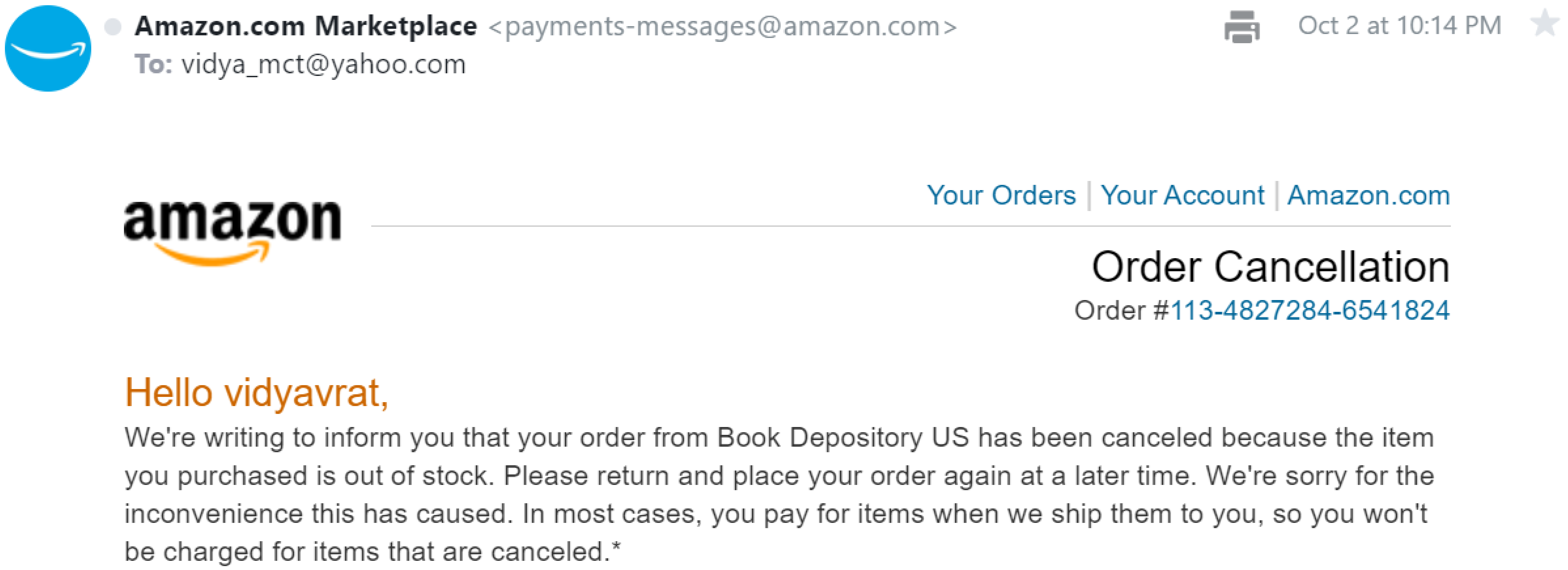
Handling Unpredictable Failures


Watch It Again


Everyone's Watching
Goldie & Bear
ChuChu TV
Barbie Dreamhouse Adventures
DreamWorks Dragons Rescue Riders
Some Assembly Required


Continue Watching for <ProfileName>

2 Second Rule



Amazon.com Marketplace <payments-messages@amazon.com>
To: vidya_mct@yahoo.com

Oct 2 at 10:14 PM

Your Orders | Your Account | Amazon.com

Order Cancellation

Order #113-4827284-6541824

Hello vidyavrat,

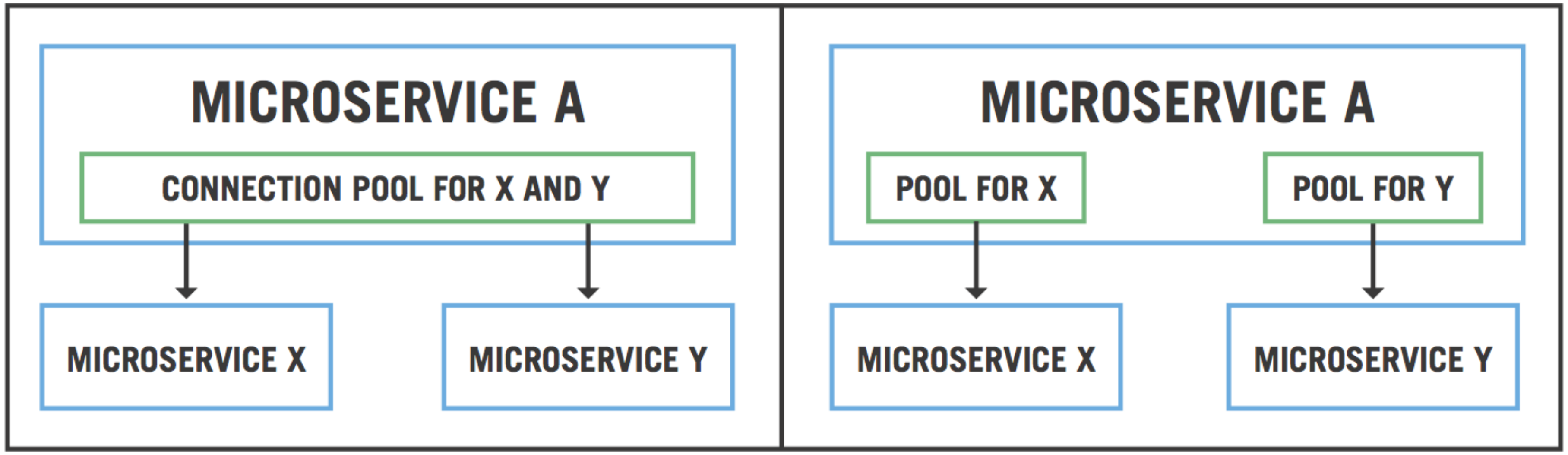
We're writing to inform you that your order from Book Depository US has been canceled because the item you purchased is out of stock. Please return and place your order again at a later time. We're sorry for the inconvenience this has caused. In most cases, you pay for items when we ship them to you, so you won't be charged for items that are canceled.*

Pattern: Bulkhead

The Bulkhead pattern is a type of application design that is "**tolerant of failure"**. In a bulkhead architecture, elements of an application are isolated into pools so that if one fails, the others will continue to function.

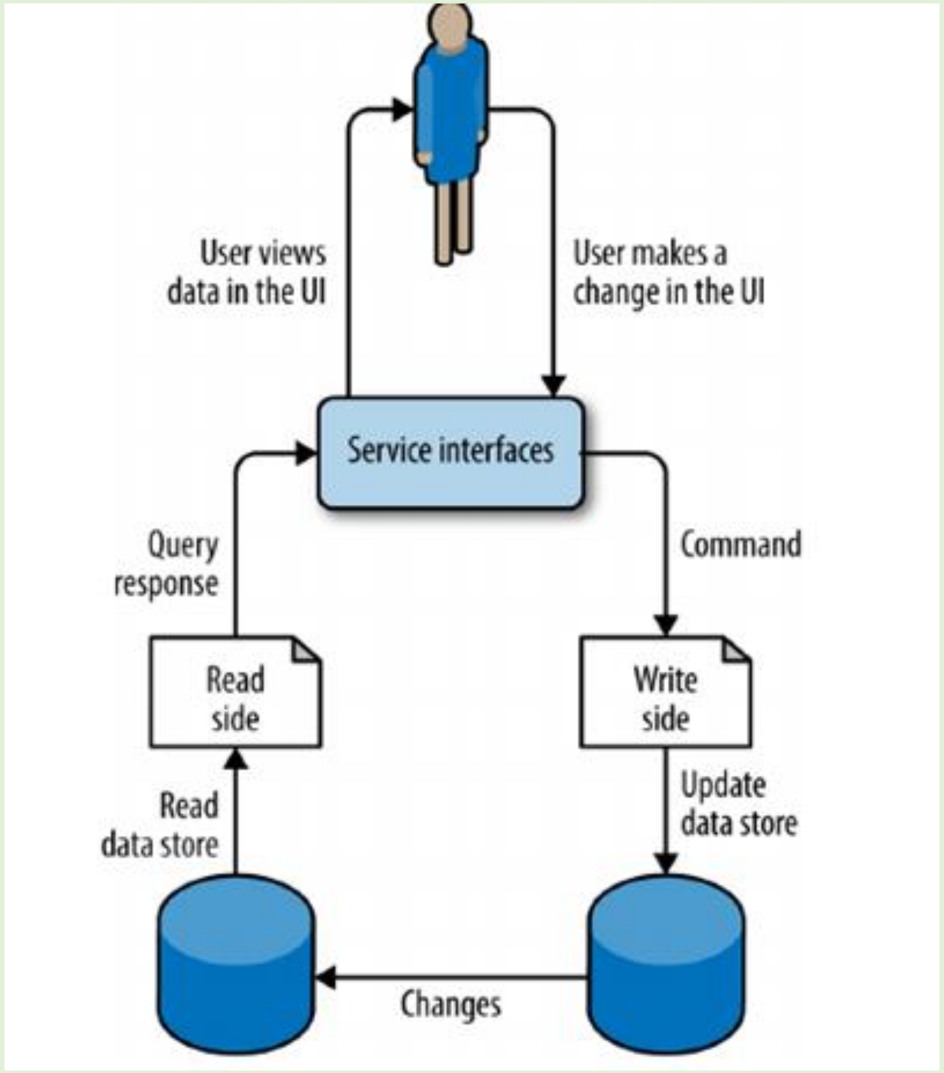
Prevents against:

- **Propagation of Failure**
- **Noisy Neighbors**
- **Unusual Demand**

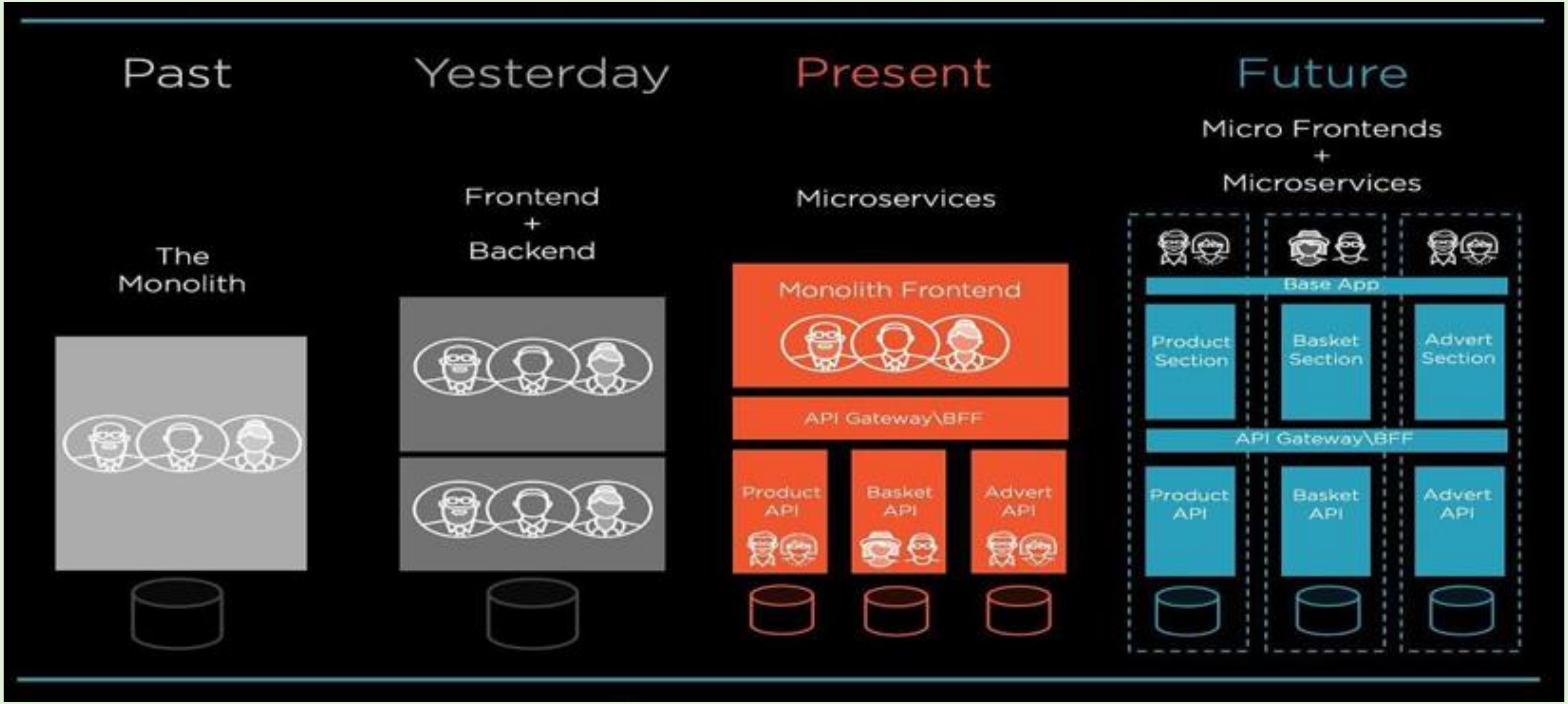


Pattern: CQRS

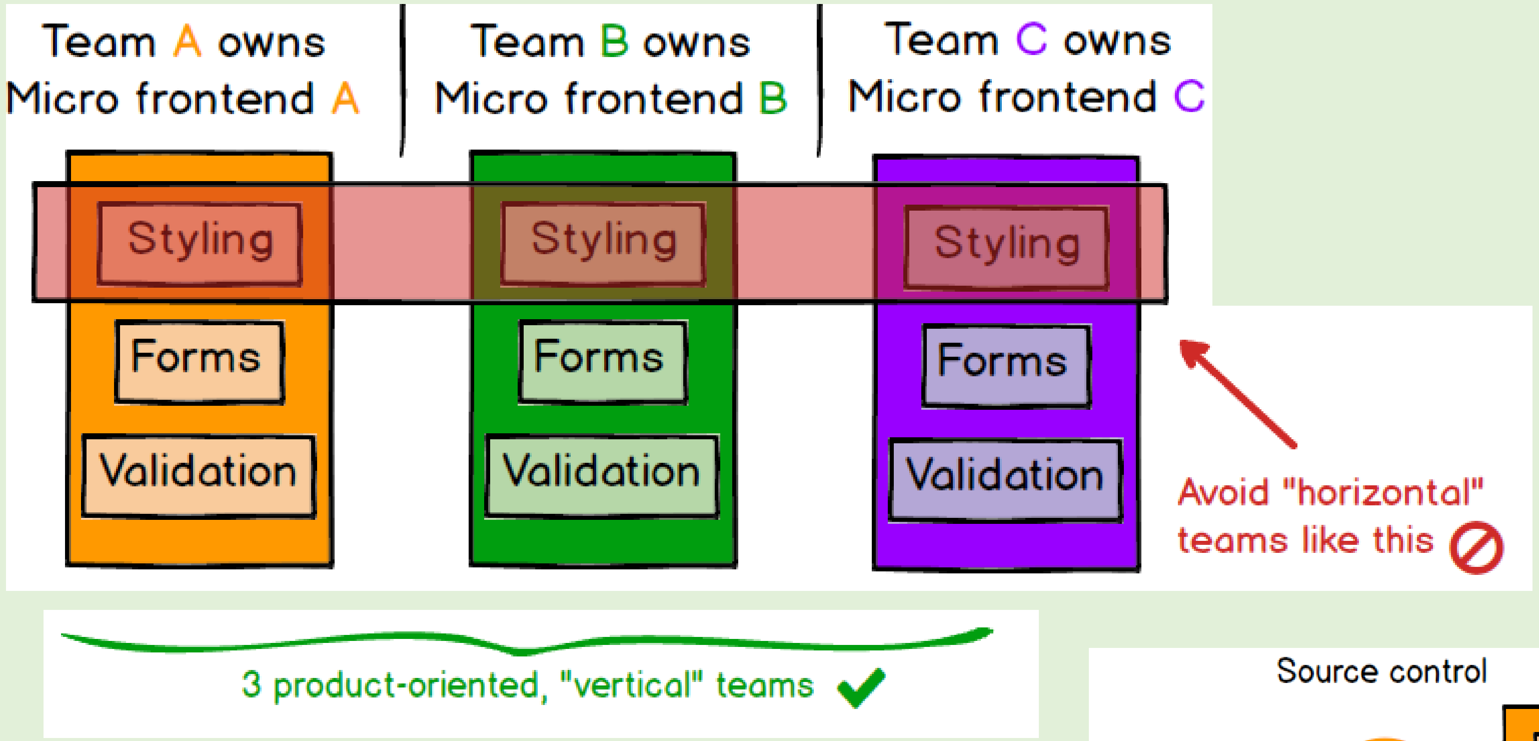
Command Query Responsibility Segregation

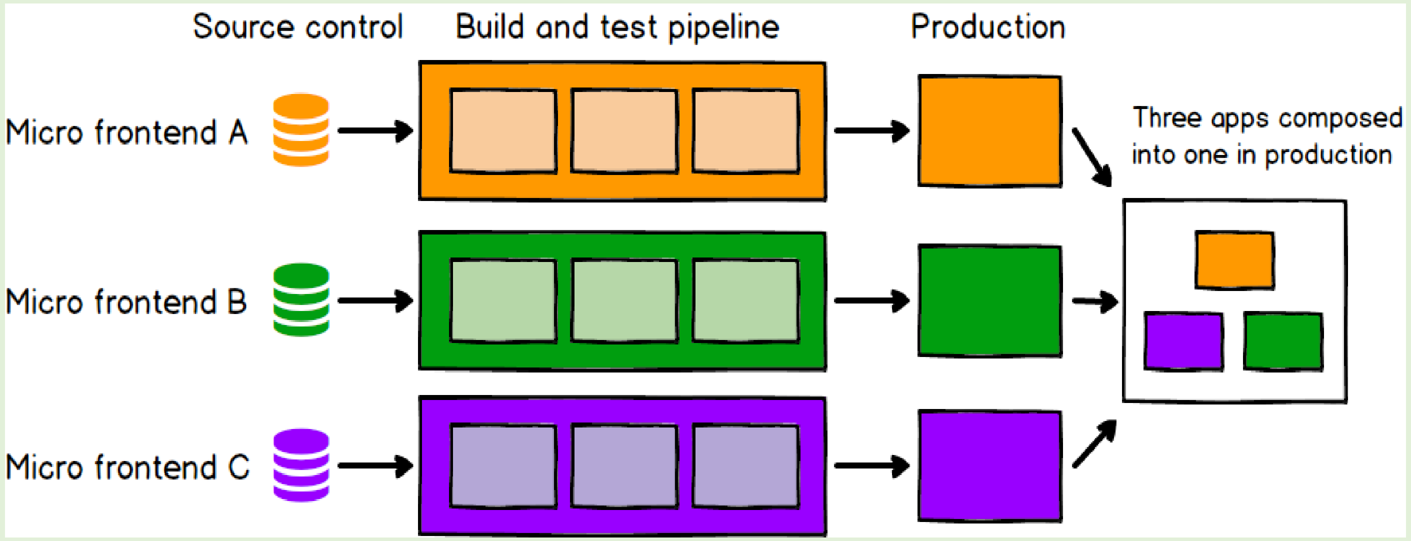


Evolutionary Architecture



Microapp Ownership





Thank You

Vidya Vrat Agarwal

www.MyPassionFor.Net | **@dotNetAuthor**
https://www.linkedin.com**/in/vidyavrat/**
https://github.com/**vidyavrat**