Microsoft

Introduction to Microservices

Microsoft Services

Conditions and Terms of Use

Microsoft Confidential

This training package is proprietary and confidential, and is intended only for uses described in the training materials. Content and software is provided to you under a Non-Disclosure Agreement and cannot be distributed. Copying or disclosing all or any portion of the content and/or software included in such packages is strictly prohibited.

The contents of this package are for informational and training purposes only and are provided "as is" without warranty of any kind, whether express or implied, including but not limited to the implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

Training package content, including URLs and other Internet website references, is subject to change without notice. Because Microsoft must respond to changing market conditions, the content should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication. Unless otherwise noted, the companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred.

Copyright and Trademarks

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

For more information, see **Use of Microsoft Copyrighted Content** at
https://www.microsoft.com/en-us/legal/intellectualproperty/permissions/default.aspx

Microsoft®, Internet Explorer®, Outlook®, SkyDrive®, Windows Vista®, Zune®, Xbox 360®, DirectX®, Windows Server® and Windows® are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Other Microsoft products mentioned herein may be either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. All other trademarks are property of their respective owners.

Agenda

- What are microservices?
- Benefits / Challenges
- Designing microservices
- Design Patterns

Microsoft

Introduction to Microservices

What are Microservices?

Microsoft Services

What are microservices?

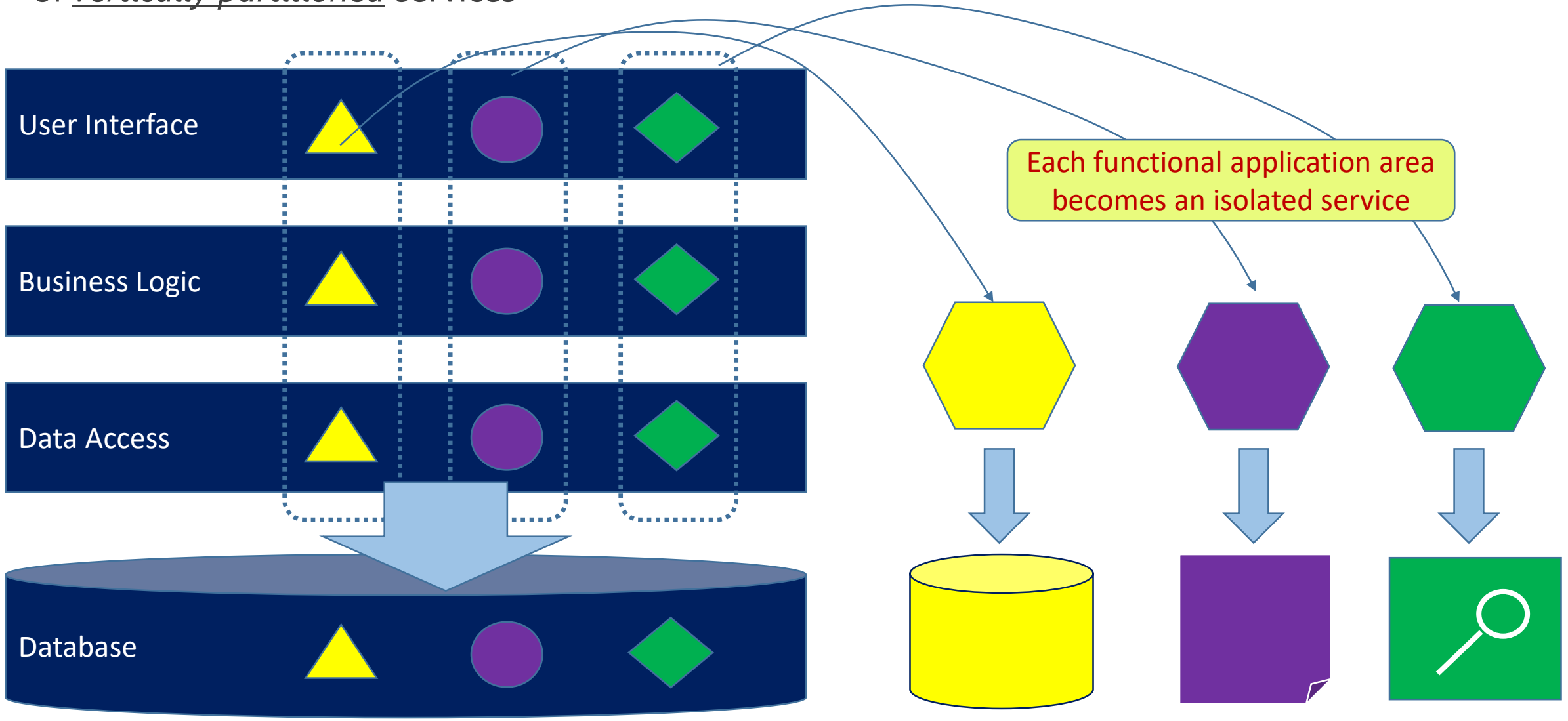
Loosely coupled services that work together, modelled around a business domain*

- Small
- Autonomous
- Scalable

* Subset of a larger domain that is capable of operating in isolation within the larger system. Typically communicates with other bounded contexts indirectly – through events and message brokers

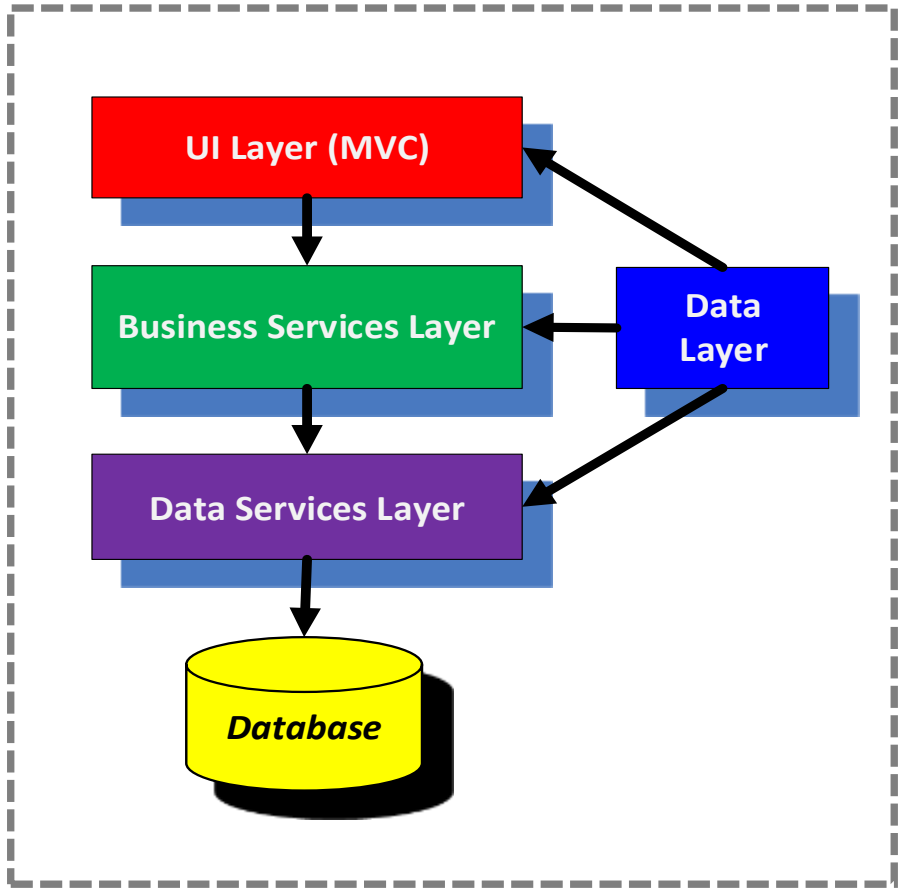
What are microservices?

- New architectural style in which app becomes of *vertically partitioned* services

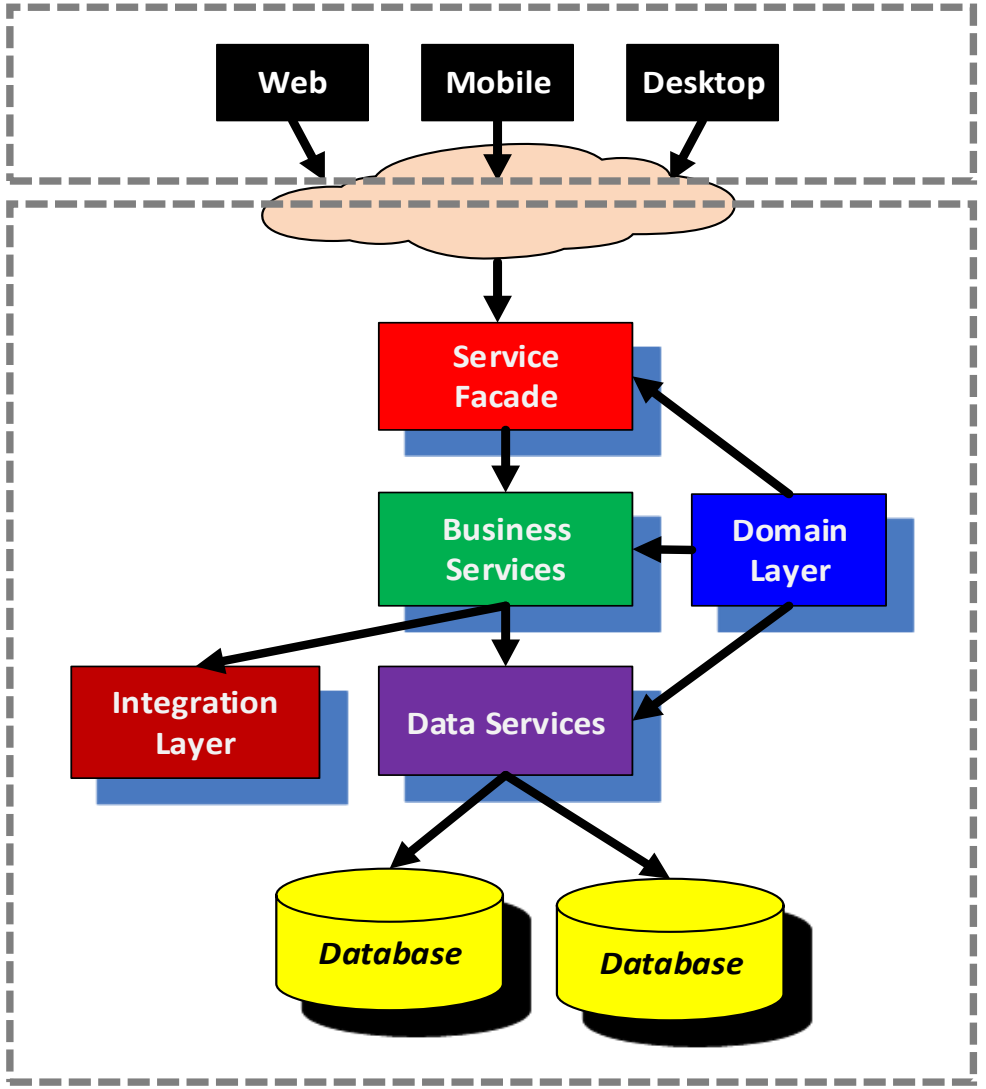


Common Monolithic Architectures

- Functional components **grouped** together horizontally



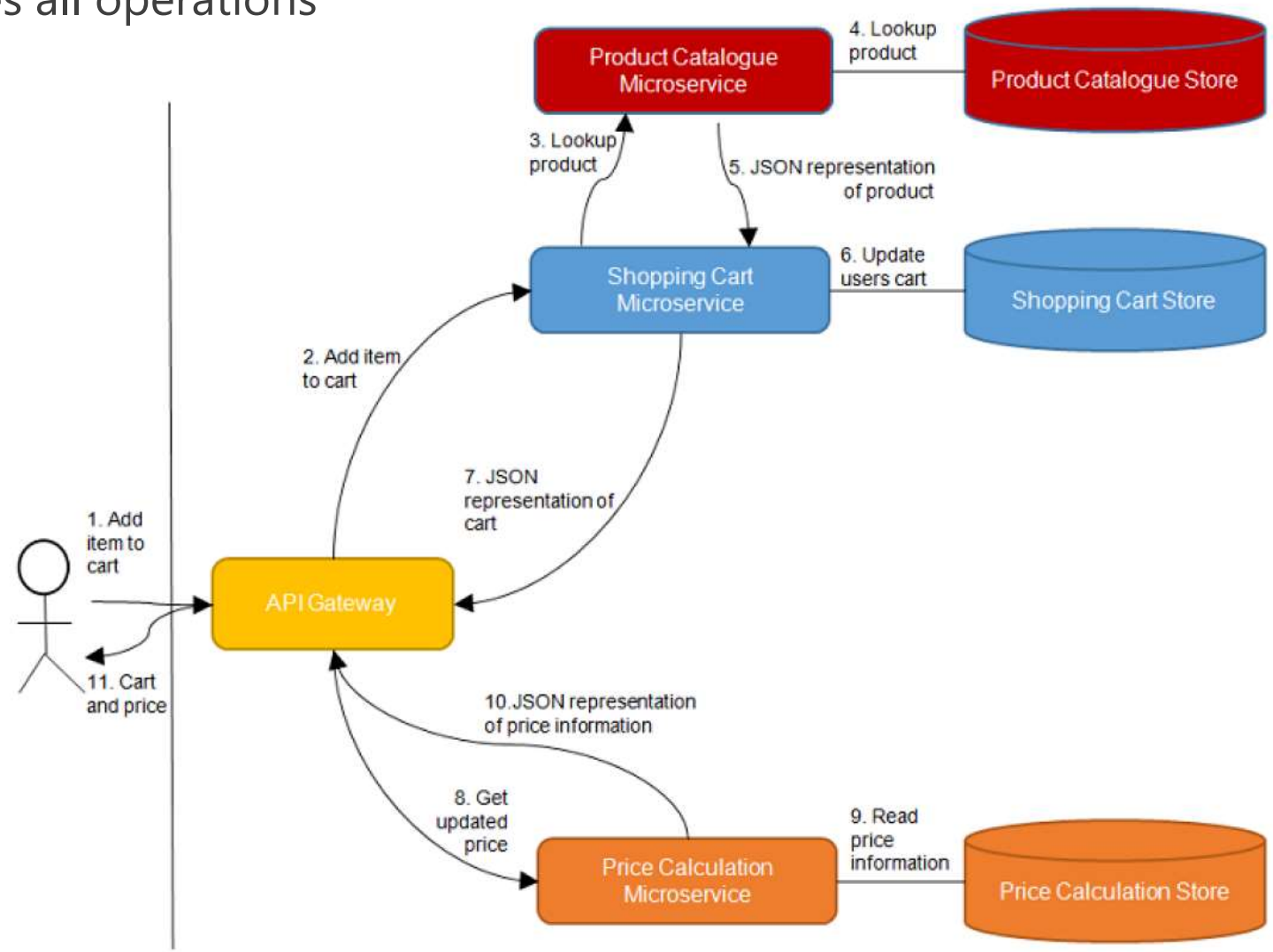
Typical Monolithic MVC App



Typical Monolithic API App

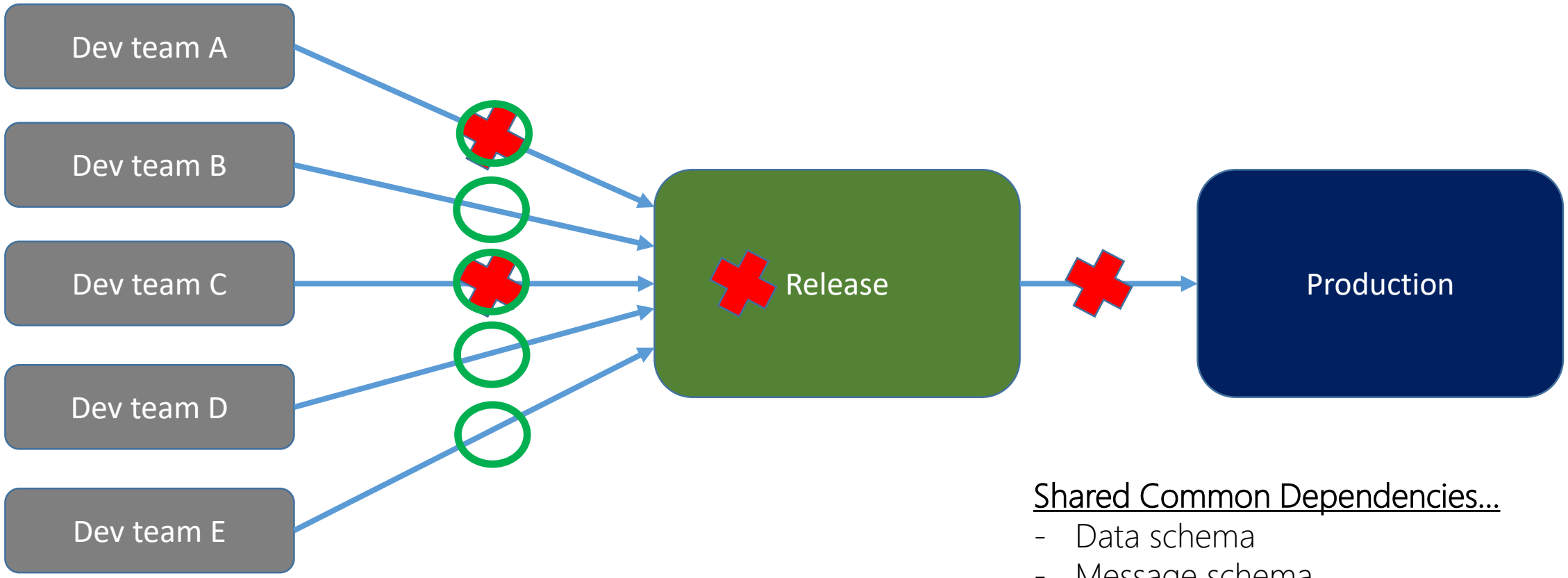
Microservices Architecture Detail View

- Each business function is a separate service with an isolated data store
- Gateway aggregates all operations



Why build microservices? Why not continue with monoliths?

- Tight coupling is the problem…



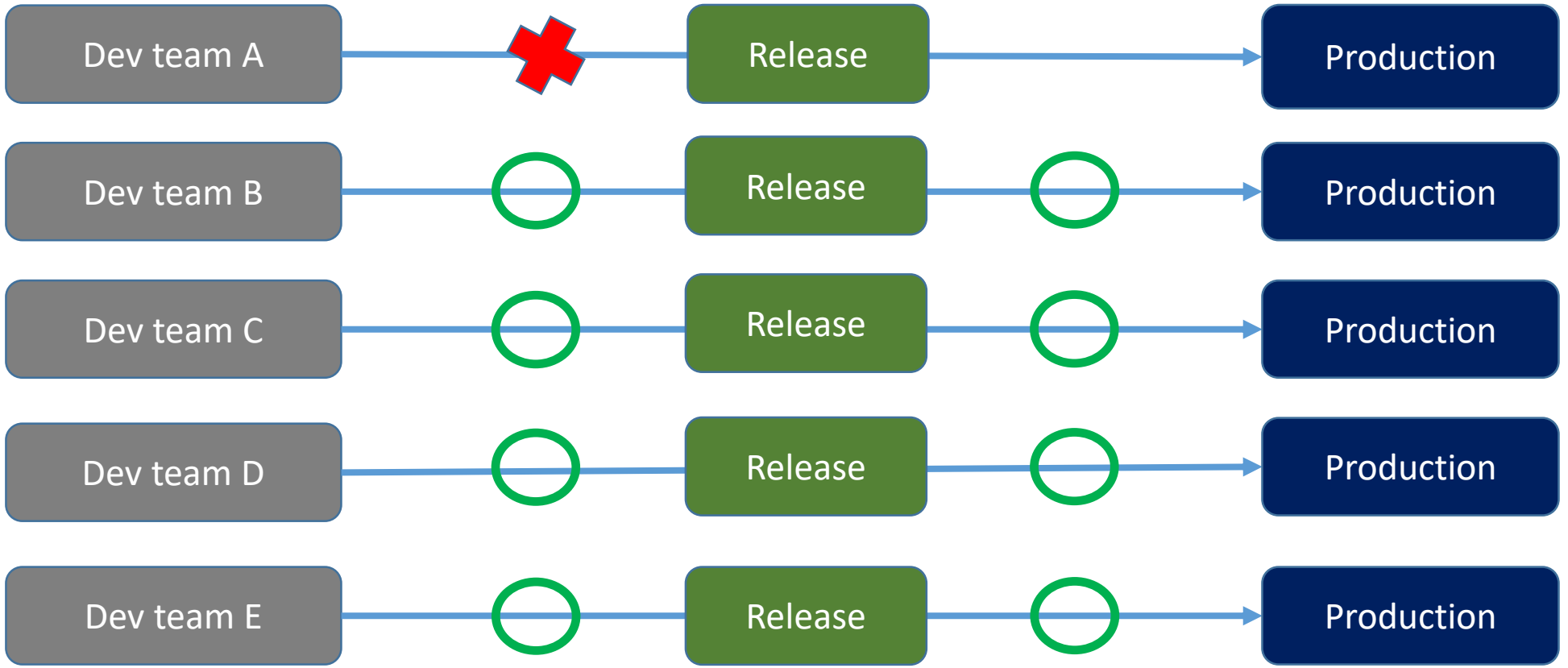
- All teams share common dependencies
- A defect in a dependency can block multiple teams

Shared Common Dependencies…

- Data schema
- Message schema
- Leaking service internals via API
- Framework/Library version
- Shared component

Why build microservices? Why not continue with a monolith?

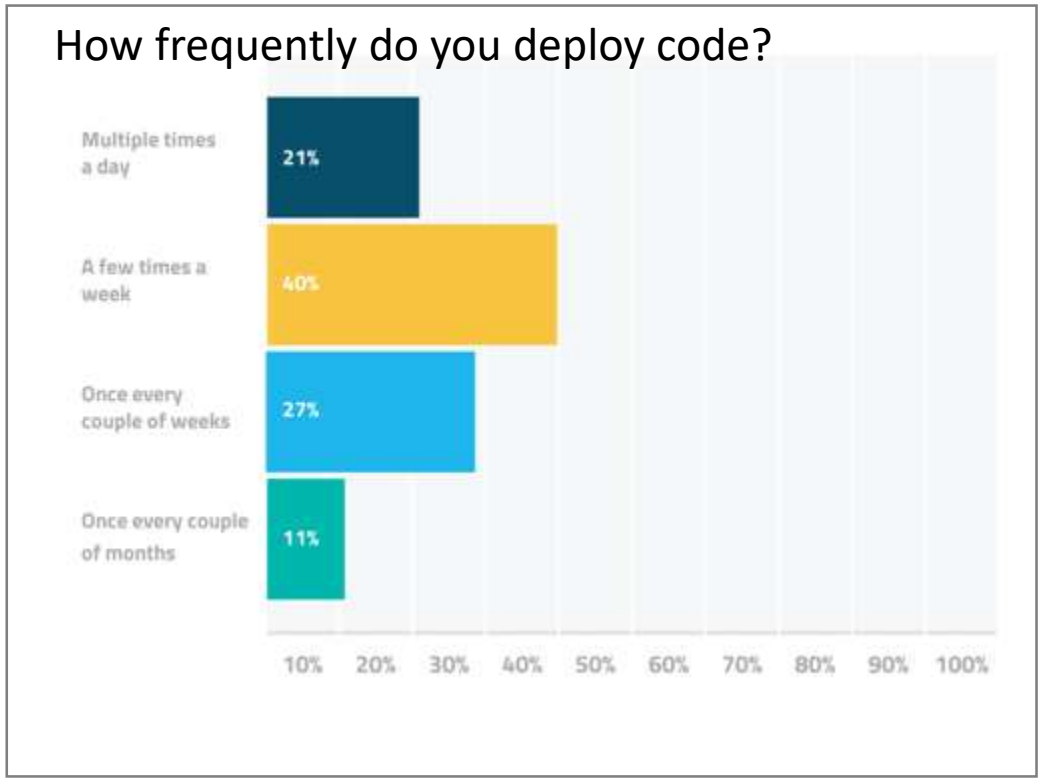
- Each team owns it own service and deploys separately…

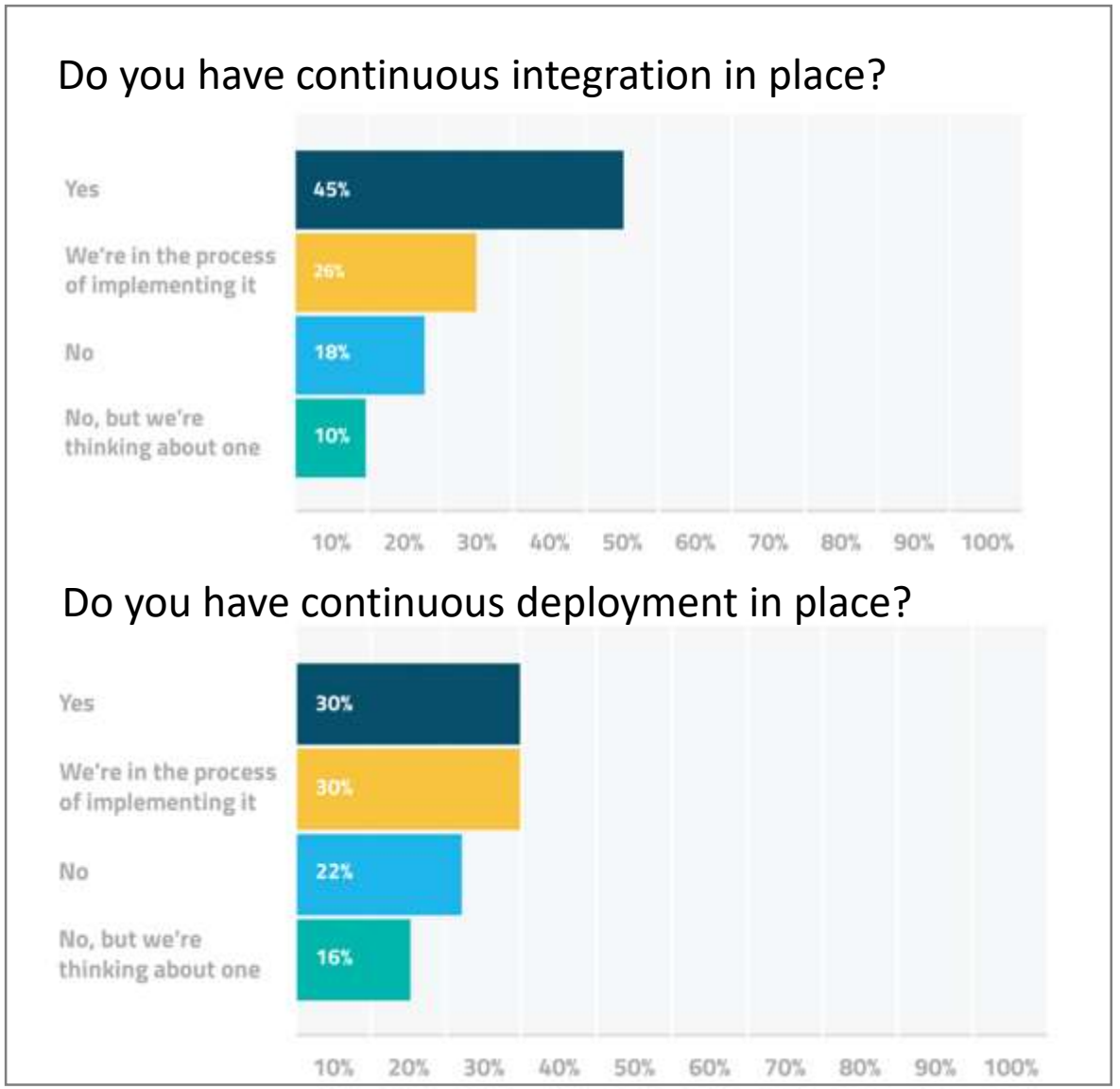


- Services do not directly share dependencies

Microservices and DevOps

- Microservices contribute to DevOps best practices and faster release cycles





Source: 2016 DevOps pulse

Microservices – case studies

Microsoft

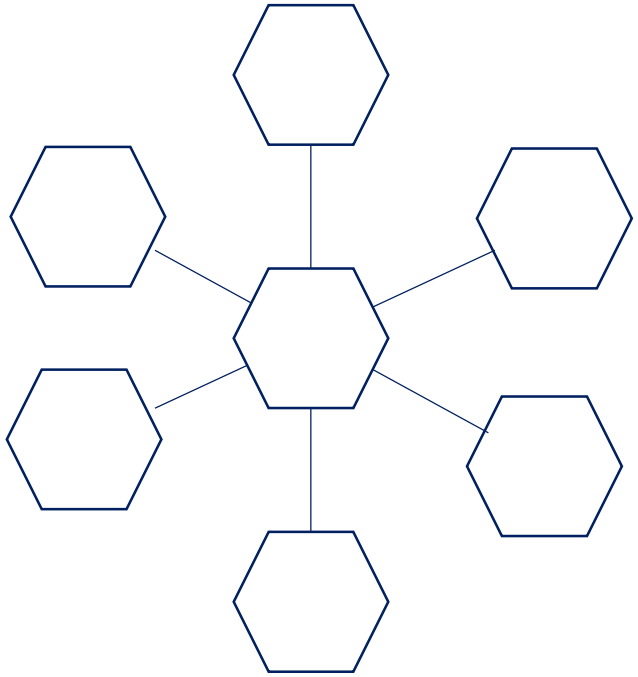
Introduction to Microservices

Benefits and Challenges

Microsoft Services

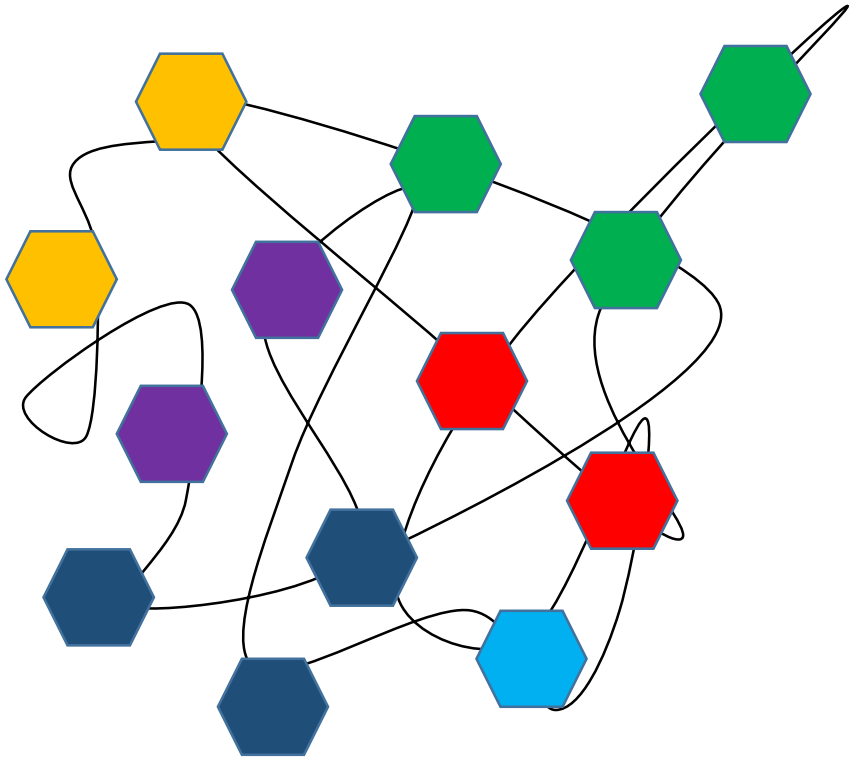
Microservices - Benefits

- Encapsulates business functionality
- Continuous innovation
- Independent deployments
- Technology diversity
- Small focused teams
- Separate scalability/availability
- Fault isolation

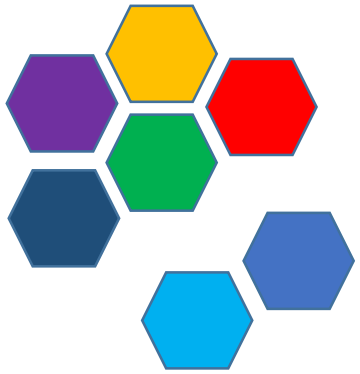


Microservices - Challenges

- Orchestration complexity
- Network congestion
- Data integrity/consistency
- Integration and versioning
- Testing
- Reliability
- Service discovery and routing
- Monitoring and logging



Modeling Microservices - Principles



- Model services around a business domain
- Make each service independently deployable
- Hide implementation details
- Data is private to its service
- Automate DevOps tasks
- Isolate failure

Less dependency more isolation!!

How can you do this in your system?

Microsoft

Introduction to Microservices

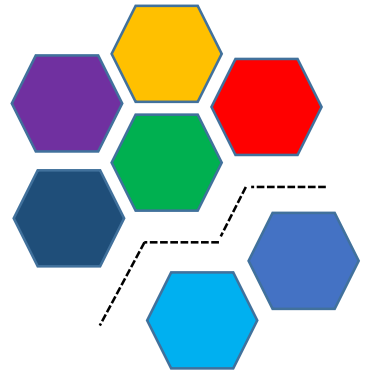
Designing Microservices

Microsoft Services

Designing Microservices

- Defining the service boundary
  - Determining the granularity of each service
- Implementing a Gateway
  - Centralizing aggregation, routing, authentication
- Implementing inter-service communication
  - Sync/async, protocol/serialization, messaging
- Data management
  - Assuring data integrity/consistency across stores
- Distributed transactions
  - Dealing with partial failure
- Monitoring services
  - Tools, frameworks, distributed tracing

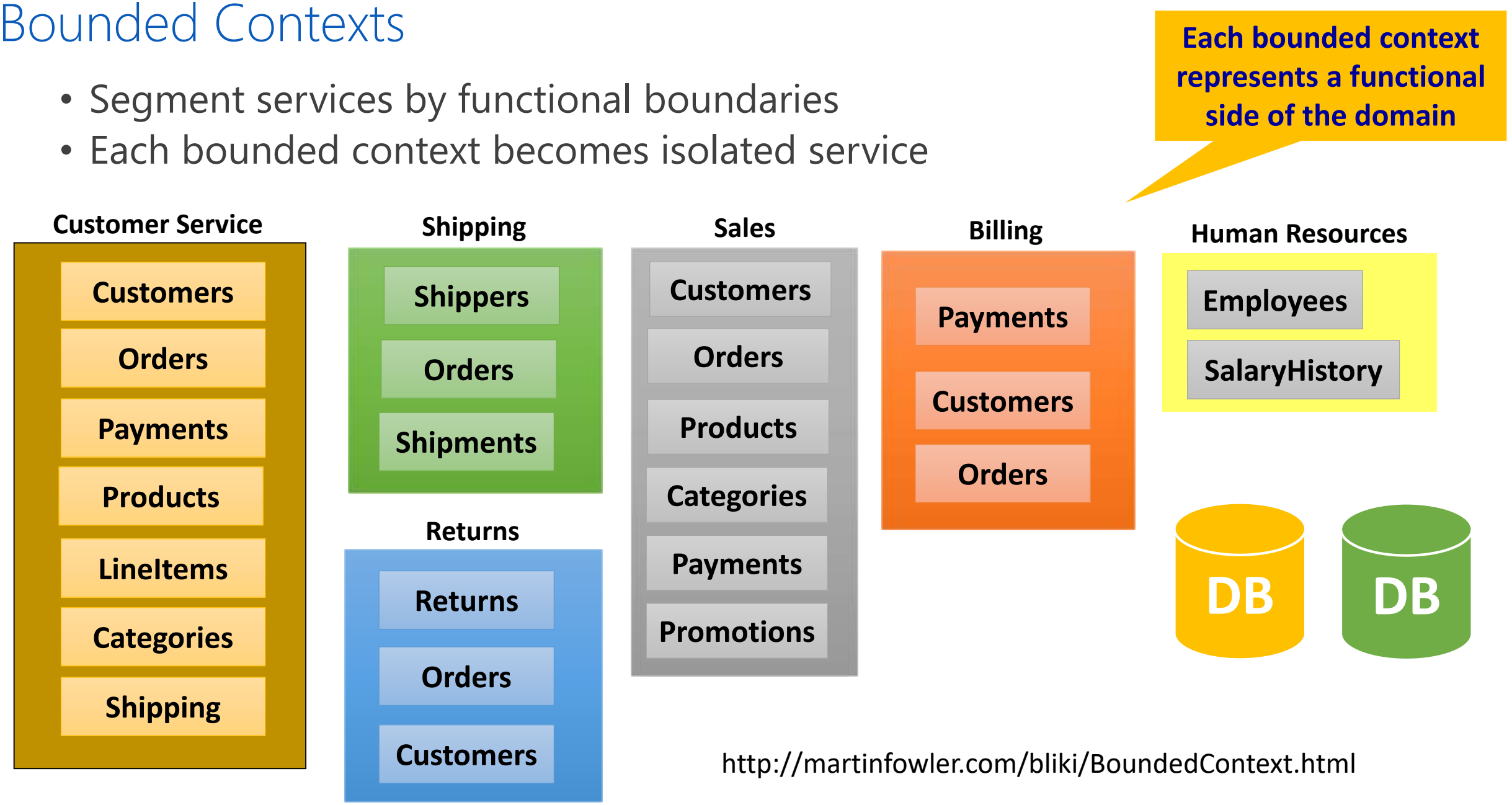
Defining the service boundary



- Start with the bounded context
- Further breakdown per non-functional requirements
- Vertical decomposition rather than horizontal (layers)
- Also consider
  - Rate of functionality change
  - Technology used
  - Communication overhead
  - Splitting data is challenging due to consistency issues
- Refactoring across boundaries is an extremely expensive operation

Bounded Contexts

- Segment services by functional boundaries
- Each bounded context becomes isolated service

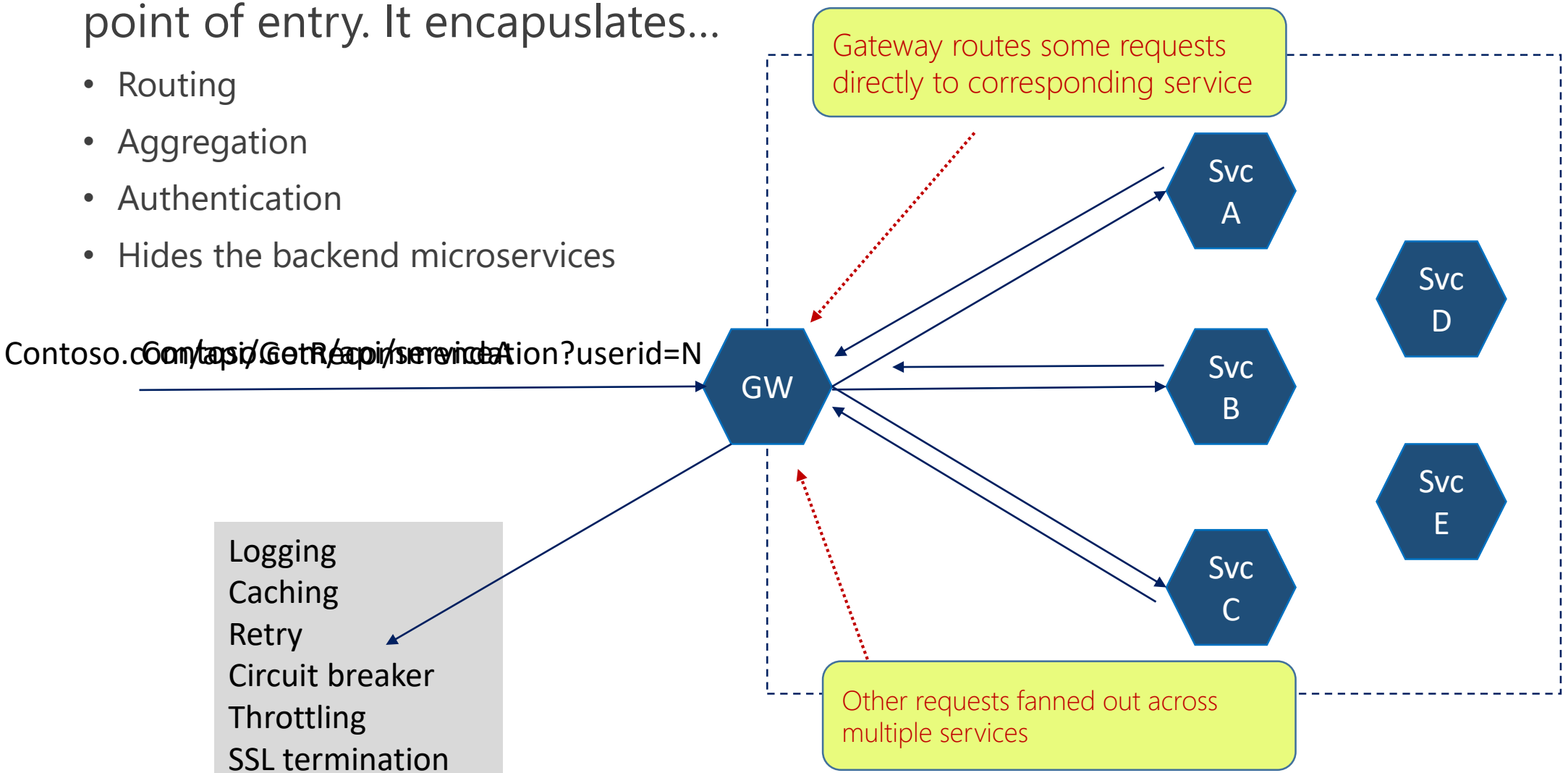


http://martinfowler.com/bliki/BoundedContext.html

API gateway

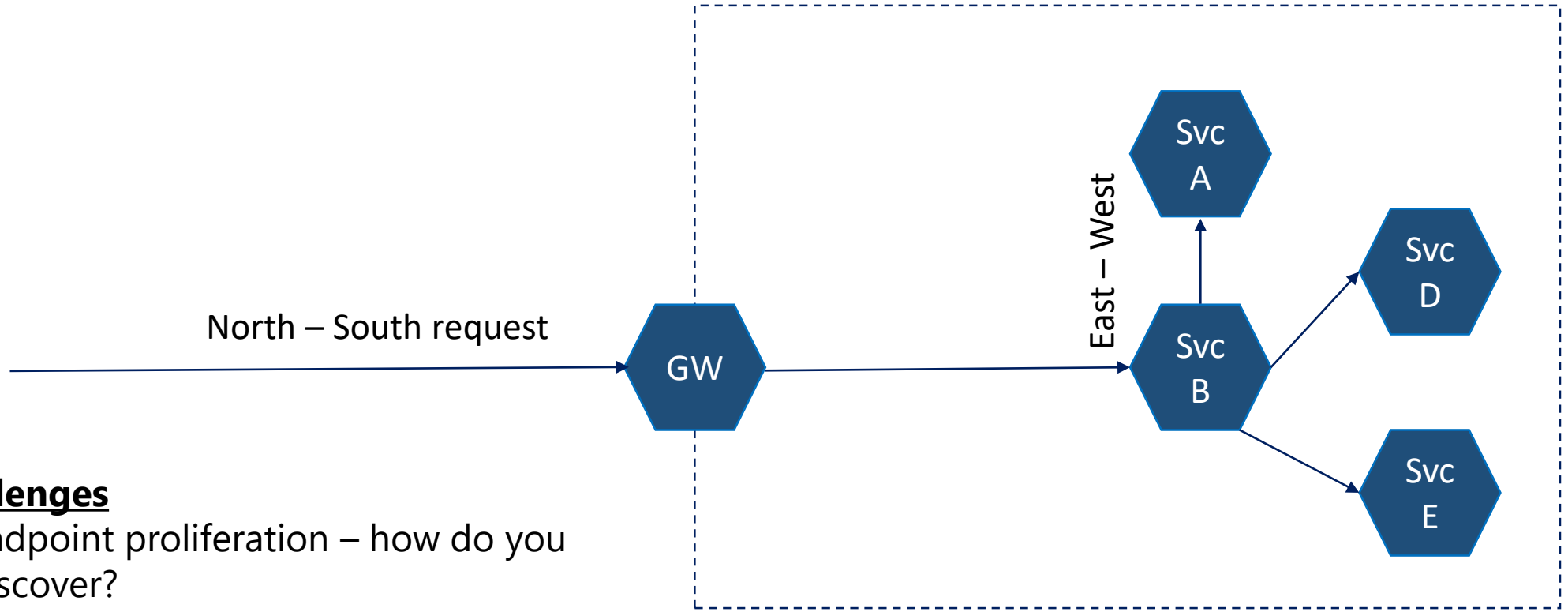
The API Gateway is the single point of entry. It encapuslates…

- Routing
- Aggregation
- Authentication
- Hides the backend microservices



Inter-Service Communication

- A significant challenge is communication across services (East-to-West)



Challenges

- Endpoint proliferation – how do you discover?
- East – West chattiness
- Overhead by serialization
- Different svc lifecycle requires decoupling
- Versioning
- IP masquerading

Microsoft

Introduction to Microservices

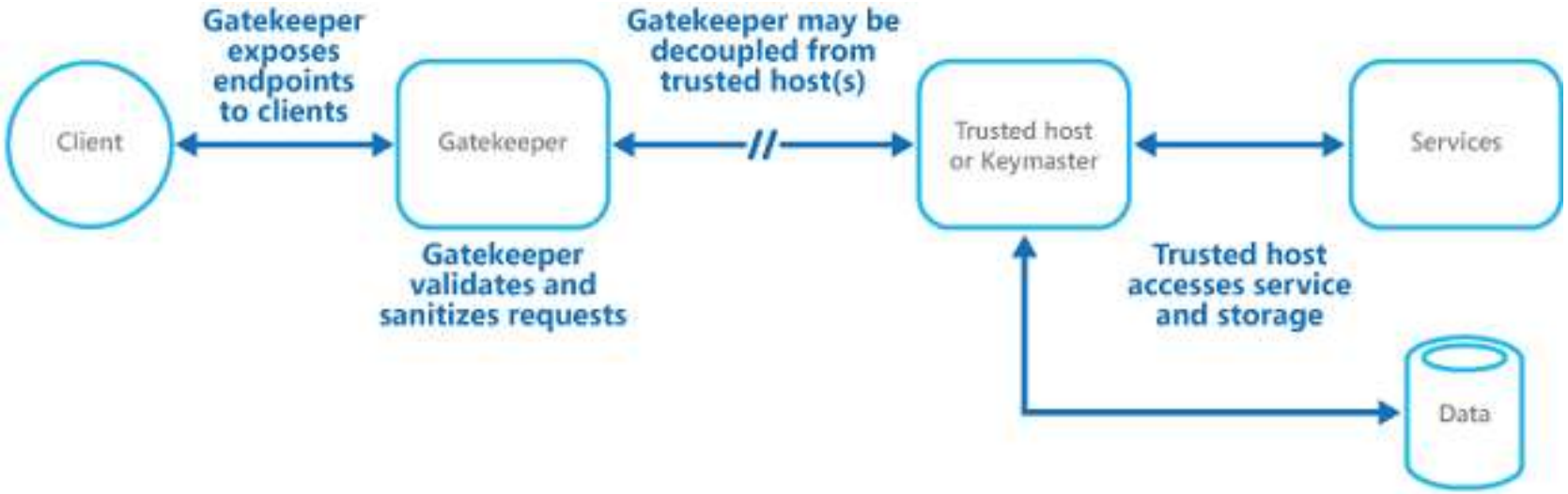
Design Patterns

Microsoft Services

Gateway Pattern

Encapsulates access to microservices, providing single point of entry for all clients…

- Exposes single URI endpoint for services
- Validates and sanitizes requests
- Aggregates operations involving multiple microservice calls
- Provides cross-cutting responsibilities: Authentication, logging, authentication, caching, monitoring
- Insulates client from internal details, service partitioning and refactorings
- Hides service discoverability and routing

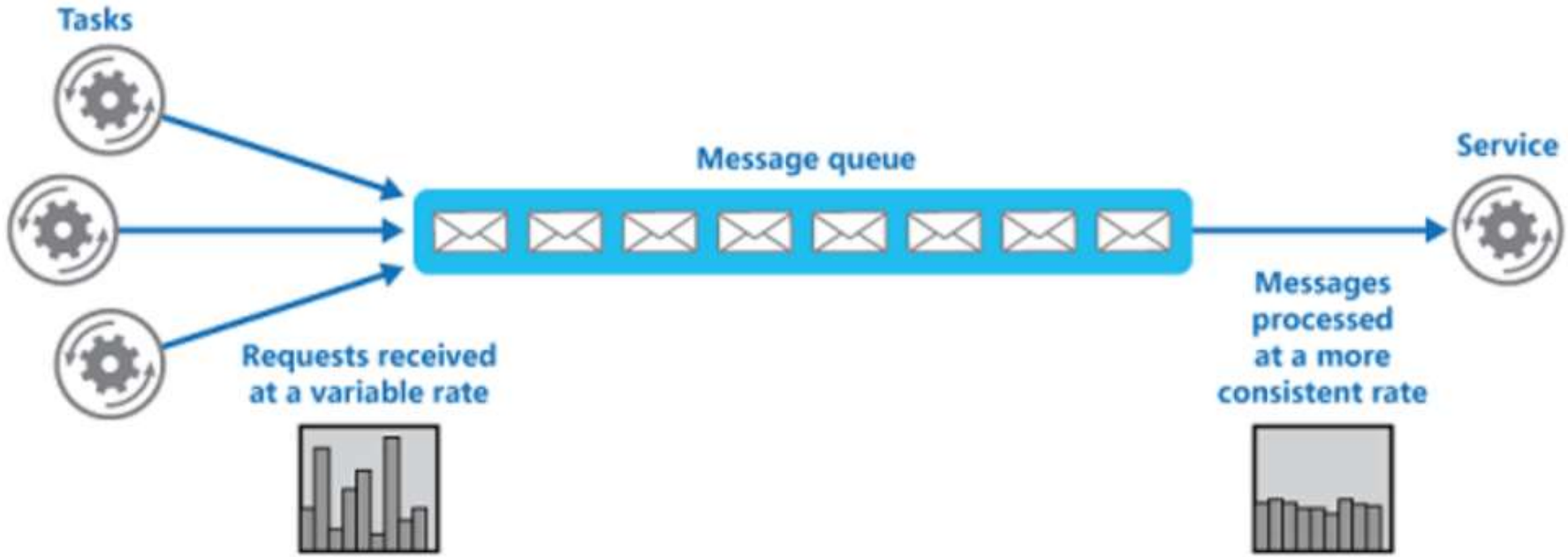


Message Broker Pattern

Implement a message queue to manage communication between two microservices…

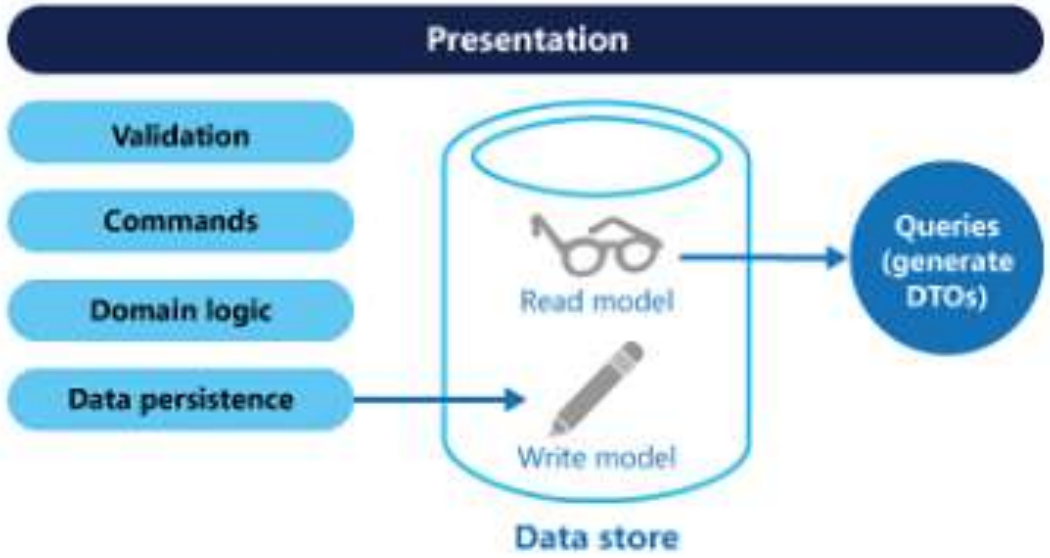
- Guarantees delivery of the message to the target service
- Smooths impact of intermittent heavy traffic that may overload the target service
- Gathers messages to be processed at later time when target service is not available

This pattern can help to minimize the impact of peaks in demand and availability and responsiveness for both the task and the service



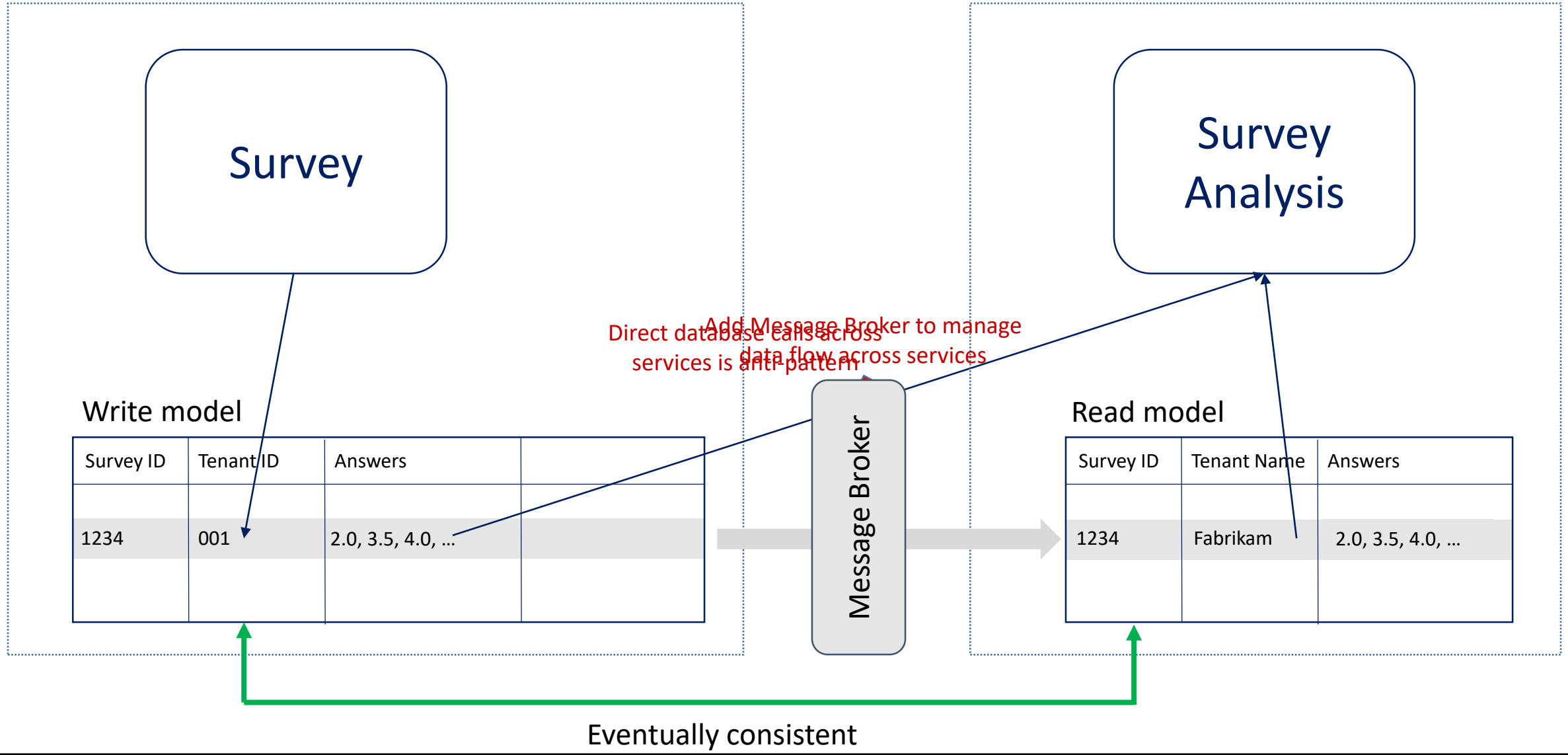
CQRS

- Knowns as the “Command and Query Responsibility Segregation Pattern”
- Segregate operations that read data from operations that update data by using separate query and update models
- Can use separate data replicas for reading and writing
- Maximizes performance, scalability, and security
- Prevents update commands from causing merge conflicts at the domain level



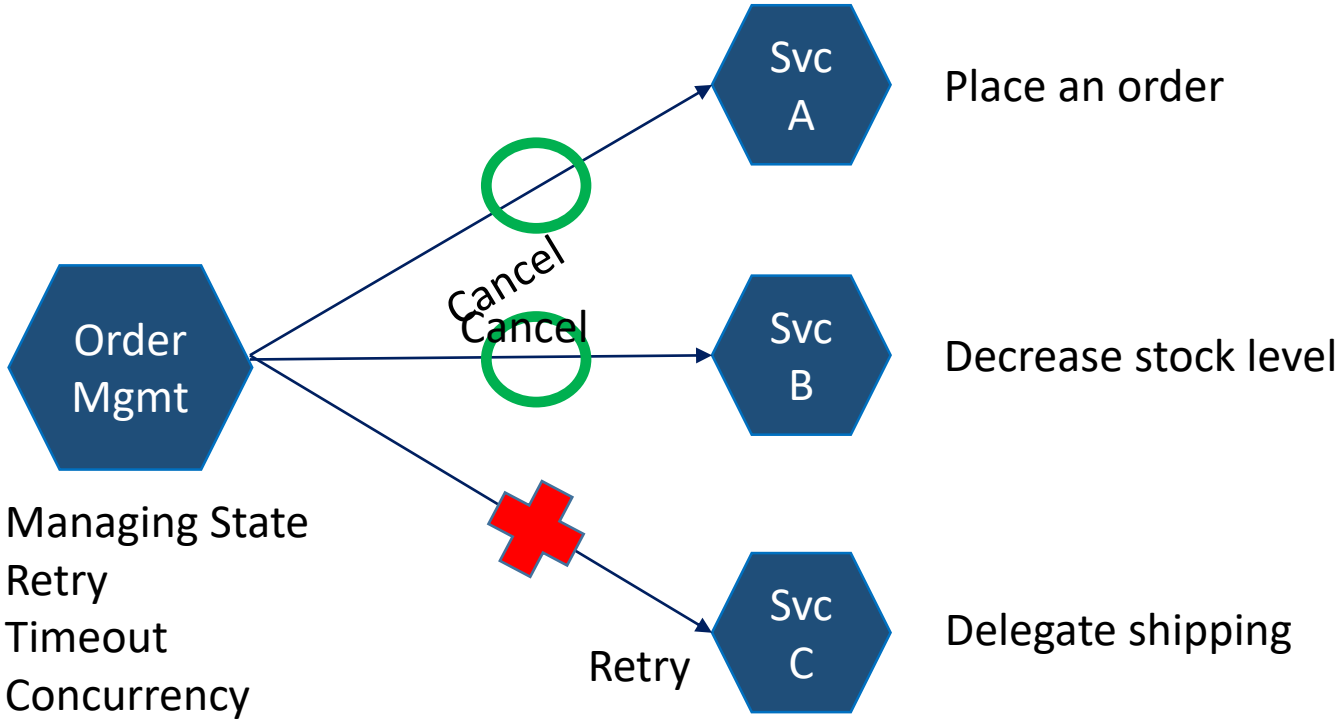
Decoupling data by CQRS

- Implementing CQRS to decouple data for read-only reporting purposes



Reversible Workflow Using Sagas

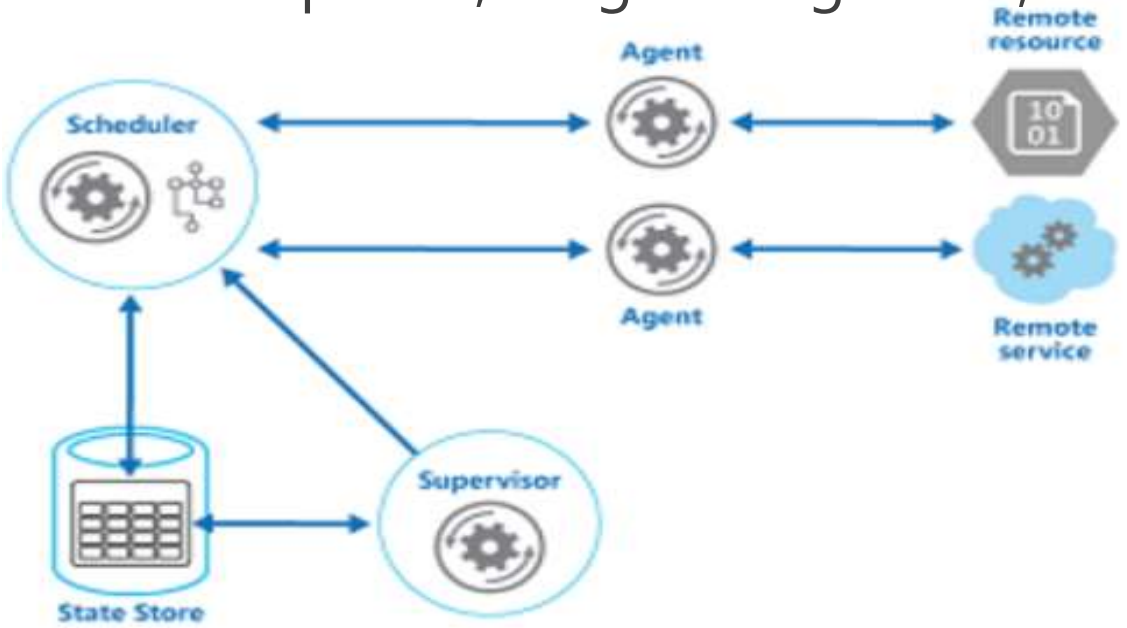
- Sagas are a long running transaction that can be written as a sequence of transactions that can be interleaved with other transactions



scheduler agent supervisor pattern

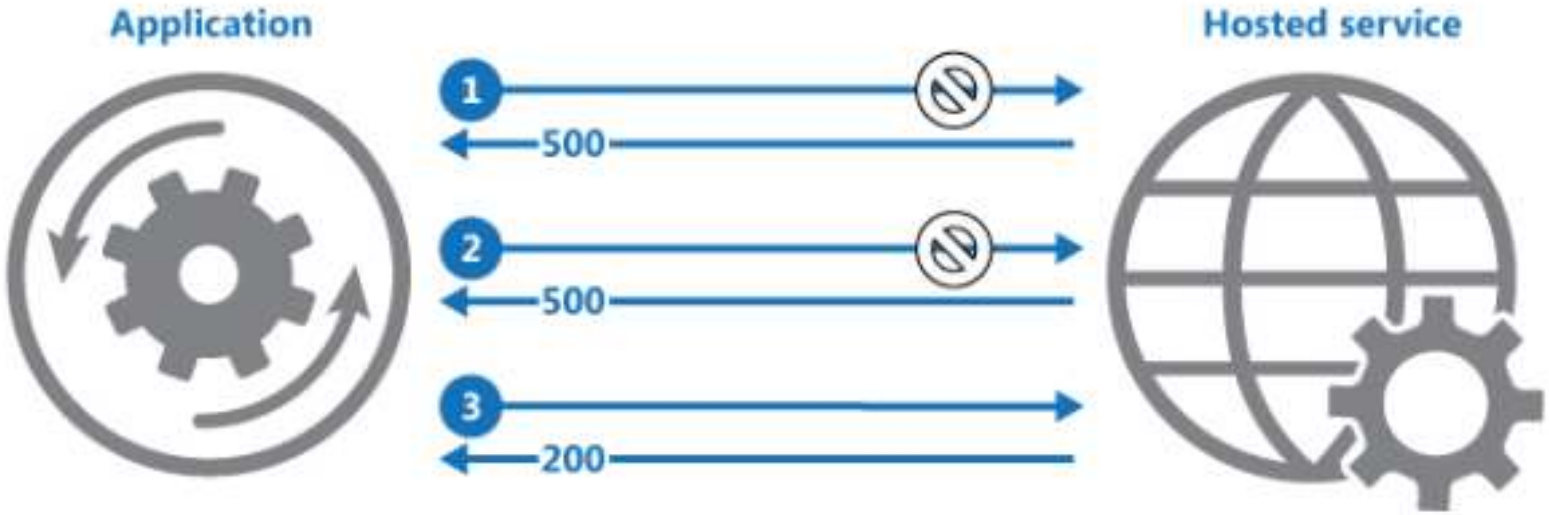
Scheduler Agent Supervisor Pattern

- Coordinates set of actions across distributed services and other remote resources
- Attempt to transparently…
  - Handle faults if actions fail
  - Or undo the effects of the work performed if the system cannot recover from a fault.
- Adds resiliency to a distributed system by enabling it to recover and retry actions that fail due to transient exceptions, long-lasting faults, and process failures



Retry Pattern

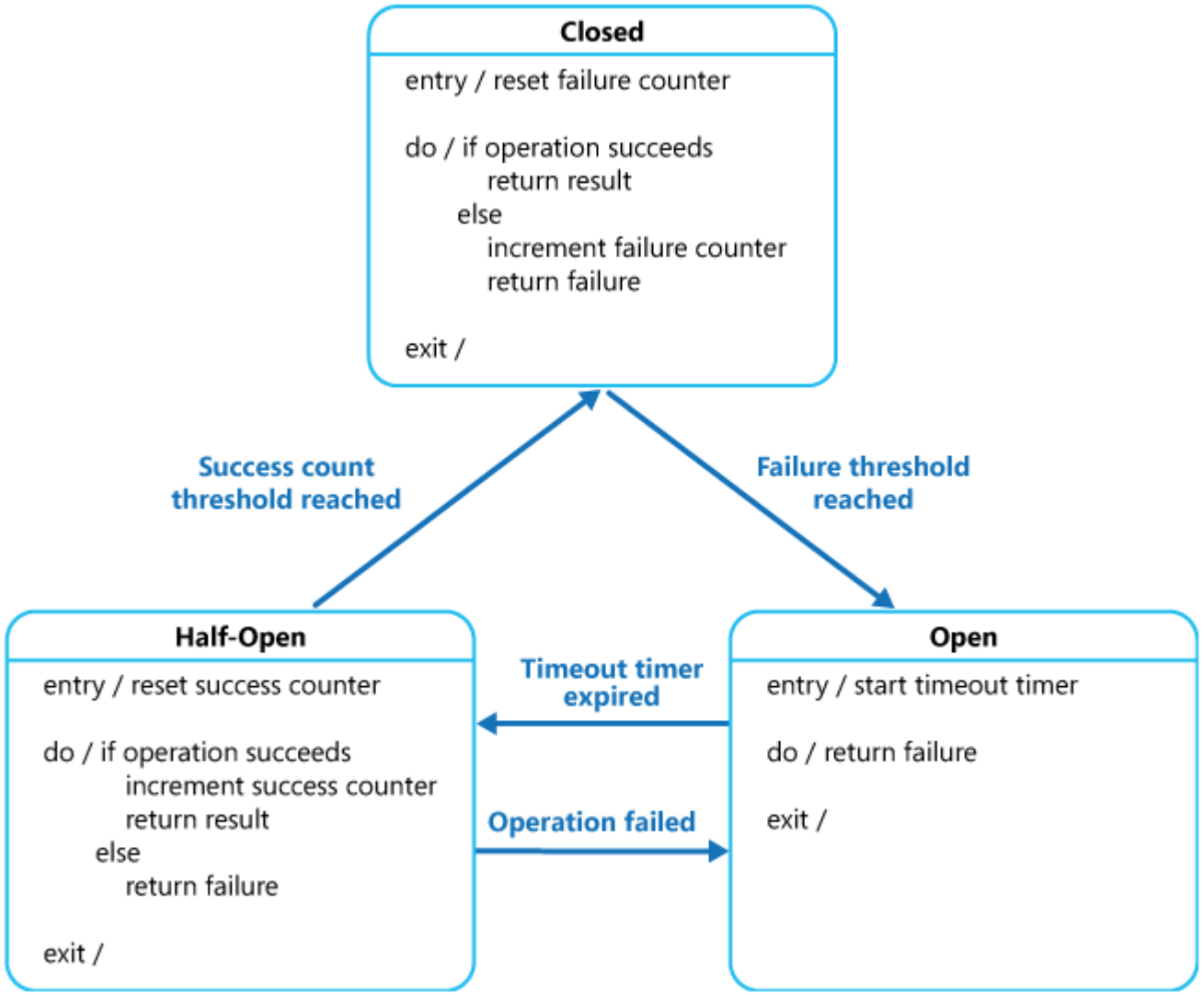
- Enables a calling service to handle temporary failures (transient faults) when connecting to other services or resources
- Transparently retries the operation for fixed number of times
- Improve the stability of the application



1. Calling service invokes operation on another service that fails with an HTTP 500 status code
2. Calling service waits for short interval and tries again. However, still fails
3. Calling services waits for longer interval and tries again. Request succeeds with an HTTP 200 status code

Circuit Breaker Pattern

- Most platforms do not provide an automated way of reconnecting to a service that moves or goes down
- The client needs to have a way or retrying and know when to stop retrying
- https://msdn.microsoft.com/en-us/library/dn589784.aspx



Options to implement microservices on Azure

- Service Fabric
- Azure Container Service (ACS)
- Azure Functions
- Docker cloud (supports Azure)
- Docker on a Virtual Machine
- Azure App service

Summary

- Microservices is a new architecture for decoupling large, monolithic applications into a set of independent, but related services
- While the architecture raises many challenges, it offers several benefits:
  - Encapsulates business functionality
  - Continuous innovation
  - Independent deployments
  - Technology diversity
  - Smaller, focused teams
  - Separate scalability/availability
  - Fault isolation
- Independent service deployment is the key
- Several hosting options available in Azure

Resources

- Microservices with Docker on Microsoft Azure (Trent Swanson, et al.)
- Building microservices (Sam Newman)
- Microservice architecture (Irakli Nadareishvili, et al.)
- https://www.nginx.com/blog/introduction-to-microservices/
- http://www.vinaysahni.com/best-practices-for-building-a-microservice-architecture
- http://www.grahamlea.com/2015/07/microservices-security-questions/
- Principles of Microservices by Sam Newman
- Adrian Cockcroft on InfoQ