Microservices Architecture Fundamentals

Ajay Pathak

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

https://github.com/ajaypathak

https://community.gartner.com/u/ajay_pathak/summary

Personal Information

2007

B Tech Computer Science and Engineering From Maulana Azad National Institute of Technology, Bhopal

2007

Software Developer

CSC

2011

Technical Consultant

2012

SDE III

talentica
SPREAD YOUR WINGS

2013

Solution Architect

Schlumberger

Agenda

- What is Service and Architecture patterns
- What is Microservice
- Microservices architecture Key Concepts
- Monolith to Microservices
- New Microservice Development
- Q&A

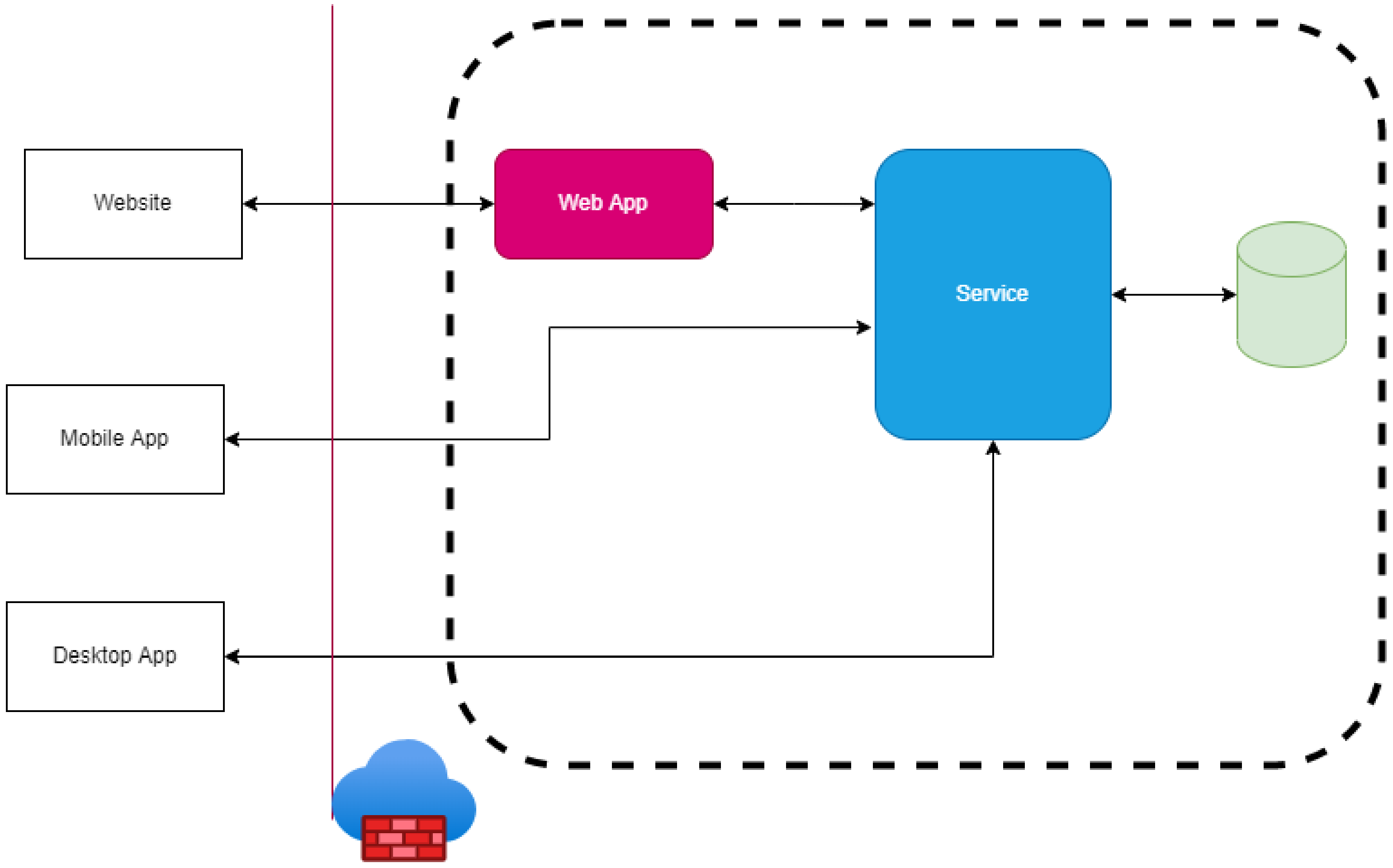
What is Service

Service encapsulates functionality and accessible to other applications and services via network

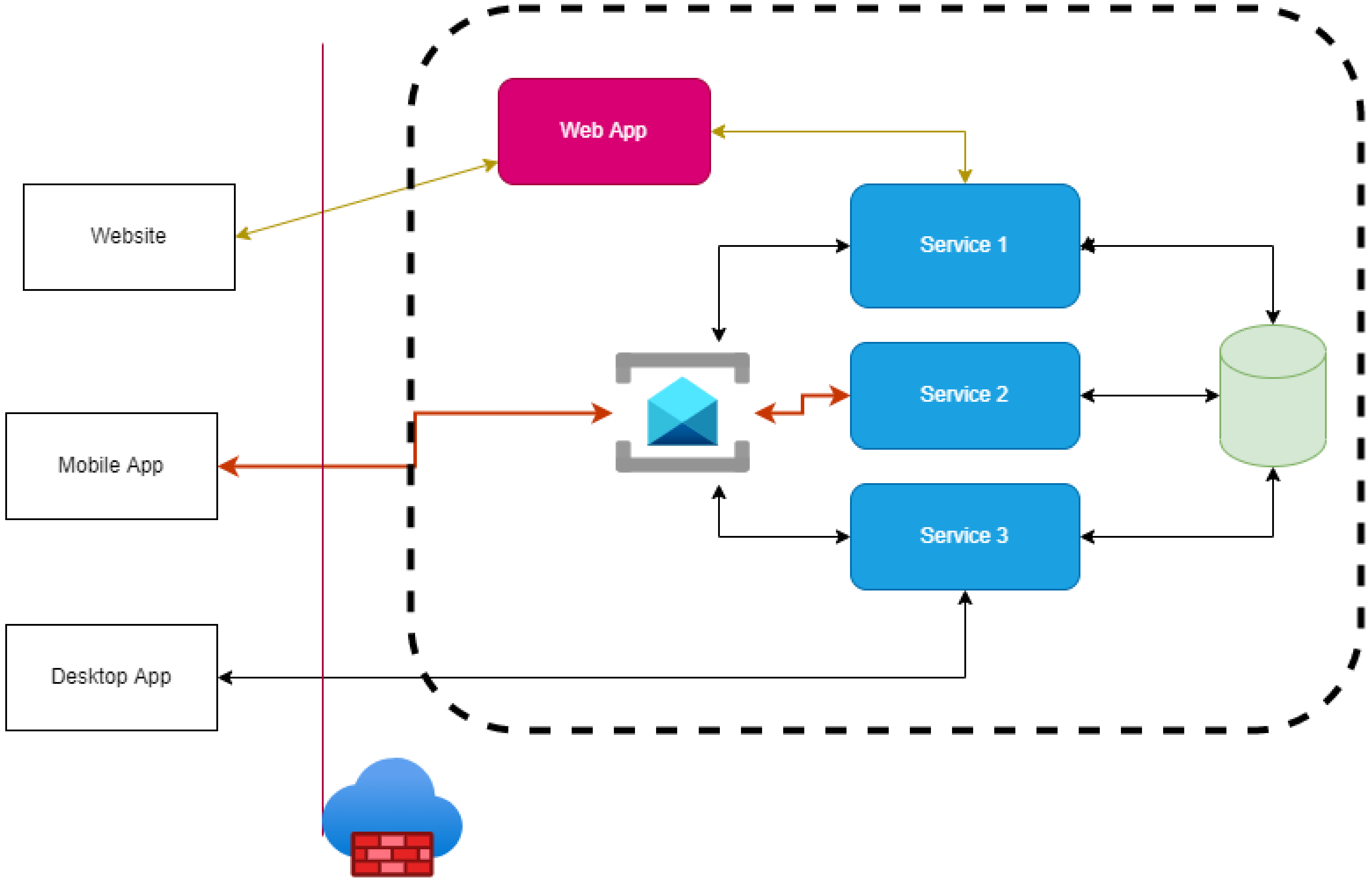
In software engineering, service-oriented architecture (SOA) is an architectural style that focuses on discrete services instead of a monolithic design. By consequence, it is as well applied in the field of software design where services are provided to the other components by application components, through a communication protocol over a network. A service is a discrete unit of functionality that can be accessed remotely and acted upon and updated independently, such as retrieving a credit card statement online. SOA is also intended to be independent of vendors, products and technologies

https://en.wikipedia.org/wiki/Service-oriented_architecture

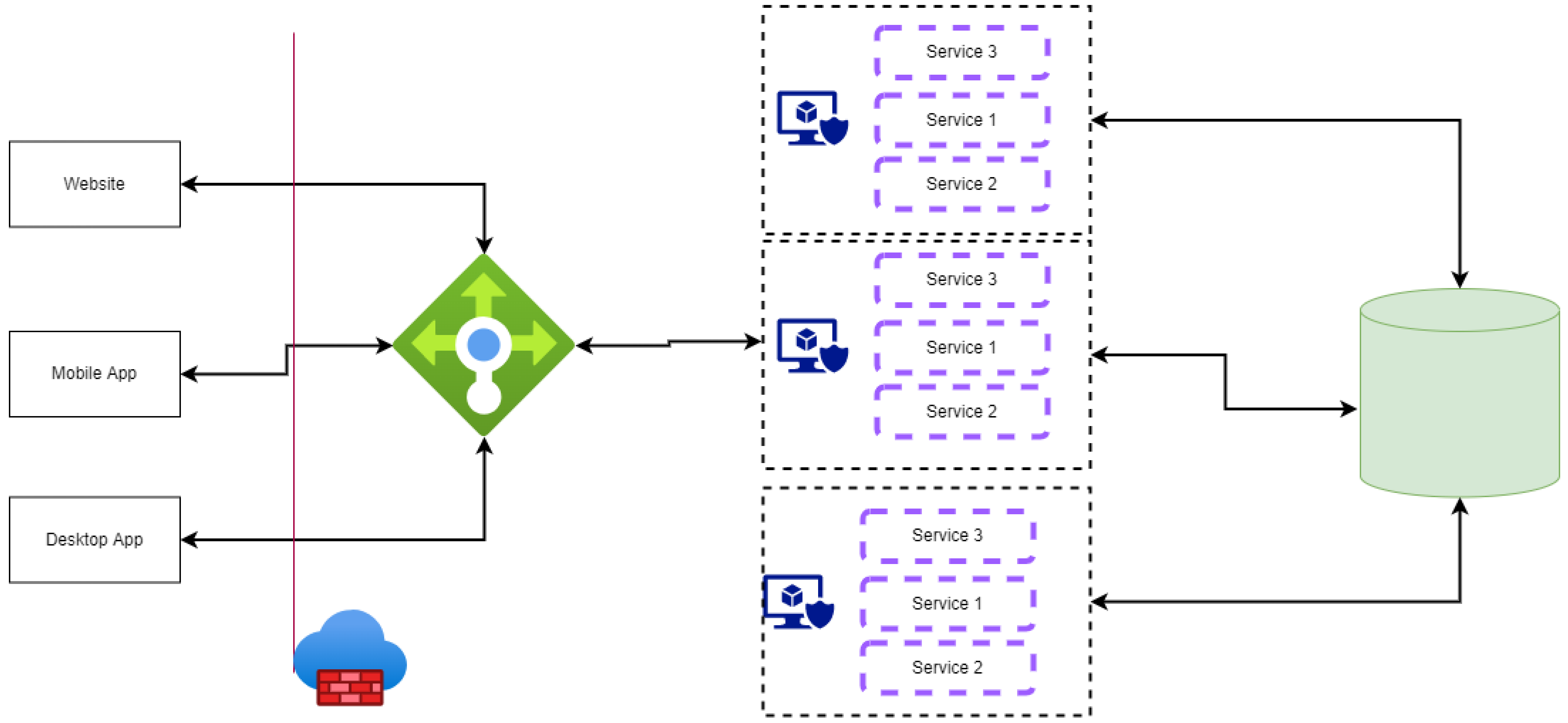
What is Service



What is Service



What is Service



Service Oriented Architecture Challenges

- Standards
- Vendor Middleware
- Lack of guidance

What Are Microservices

Microsoft

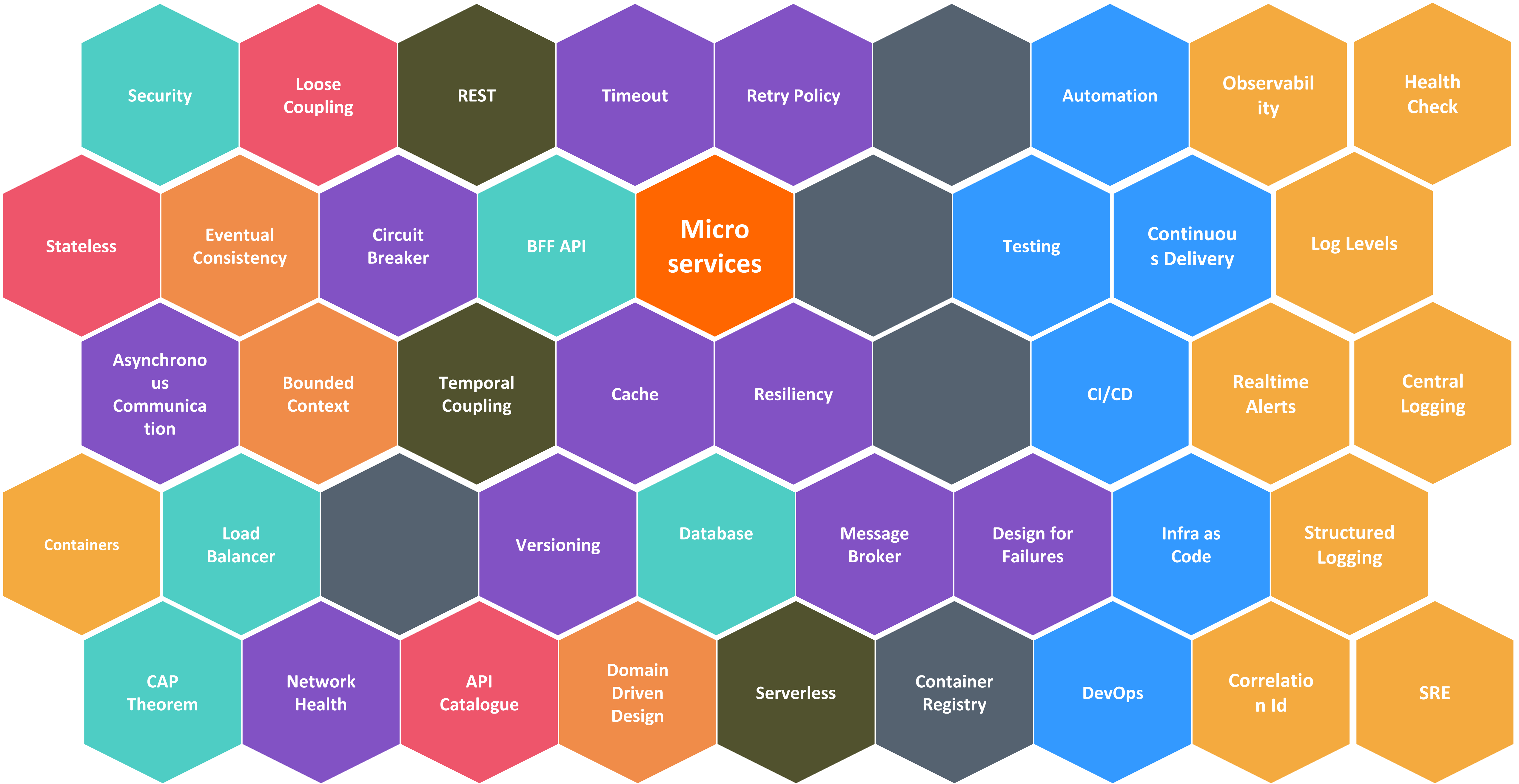
Microservices are an architectural approach to building applications where each core function, or service, is built and deployed independently. Microservice architecture is distributed and loosely coupled, so one component's failure won't break the whole app. Independent components work together and communicate with well-defined API contracts.

Google

A microservices architecture is a type of application architecture where the application is developed as a collection of services. It provides the framework to develop, deploy, and maintain microservices architecture diagrams and services independently.

Microservices Architecture

Microservices Architecture



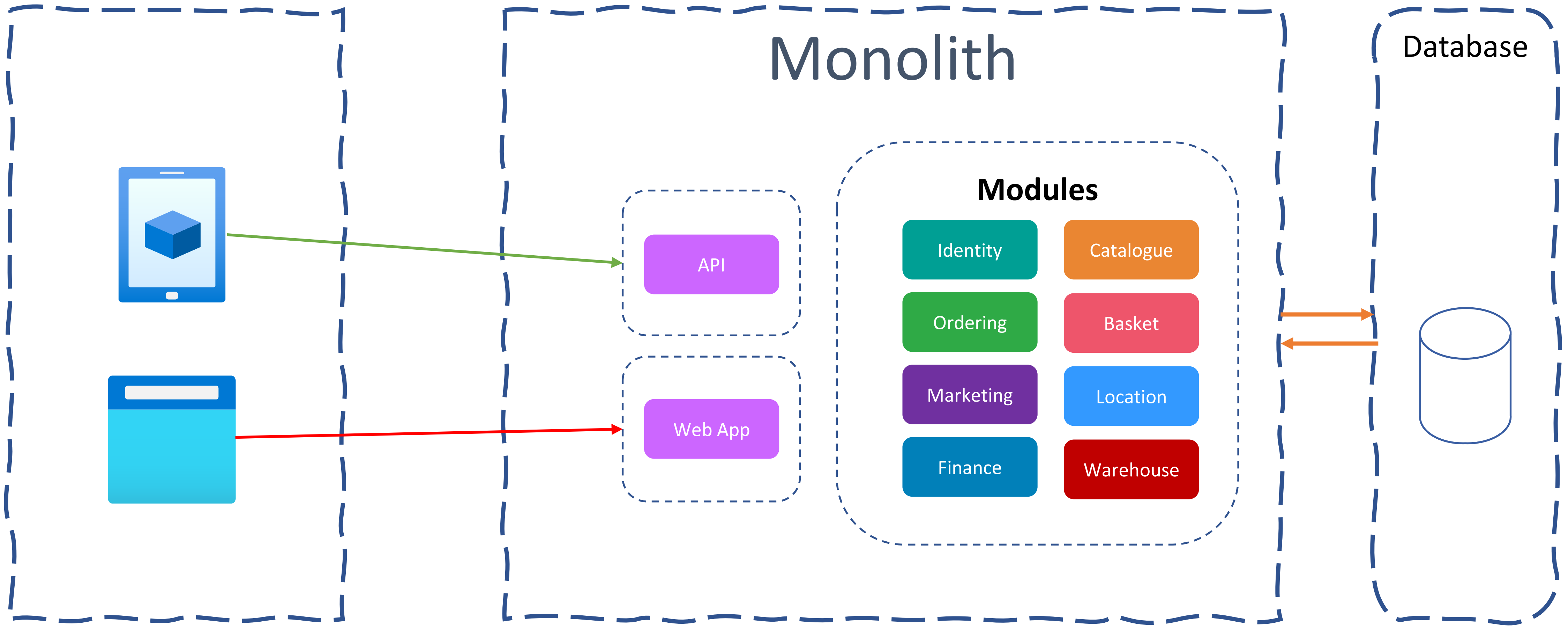
Microservices Benefits

- Technology Heterogeneity
- Robustness
- Independent Deployment
- Scalability
- Data Isolation
- Agility
- Two Pizza teams
- Falut Isolation

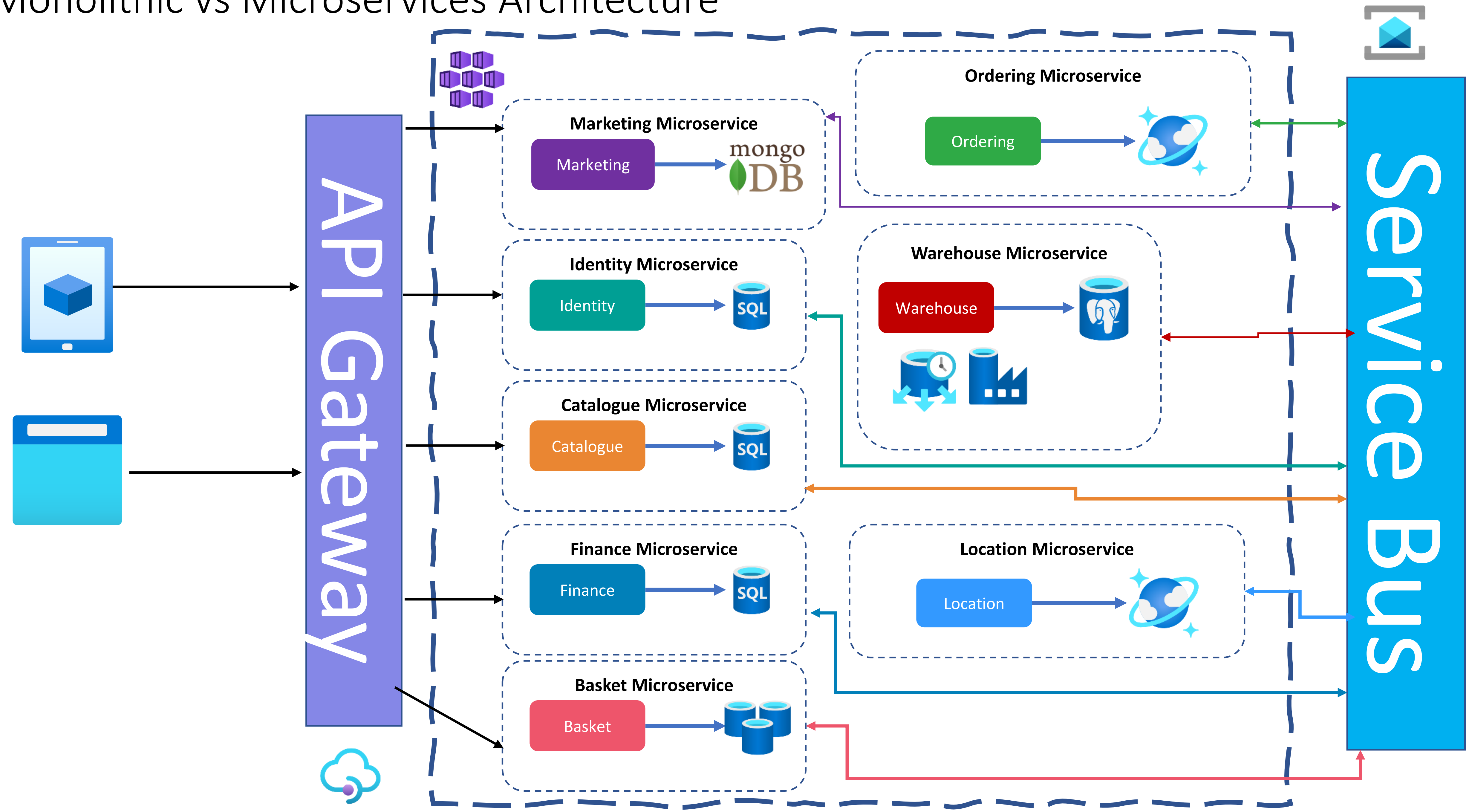
Microservices Challenges

- Technology Overload
- Monitoring & Troubleshooting
- Security
- Developer Experience
- Development Cost
- Testing
- Reporting
- Latency
- Data Consistency

Monolithic vs Microservices Architecture



Monolithic vs Microservices Architecture



Microservices Key Concepts

- Autonomous
- Domain Driven Design
- Ownership Culture
- Resiliency
- Observability
- Automation

Autonomous

- Loose Coupling
- Contracts and Interfaces
- Stateless
- Backward Compatibility
- Parallel Development
- Independently Deployable

Autonomous

- Loose Coupling
- Contracts and Interfaces
- Stateless
- Backward Compatibility
- Parallel Development
- Independently Deployable

Domain Driven Design

- Cohesion
- Bounded Context
- Coupling
- Event Storming
- Identification of Coupling
- Easily Rewritable

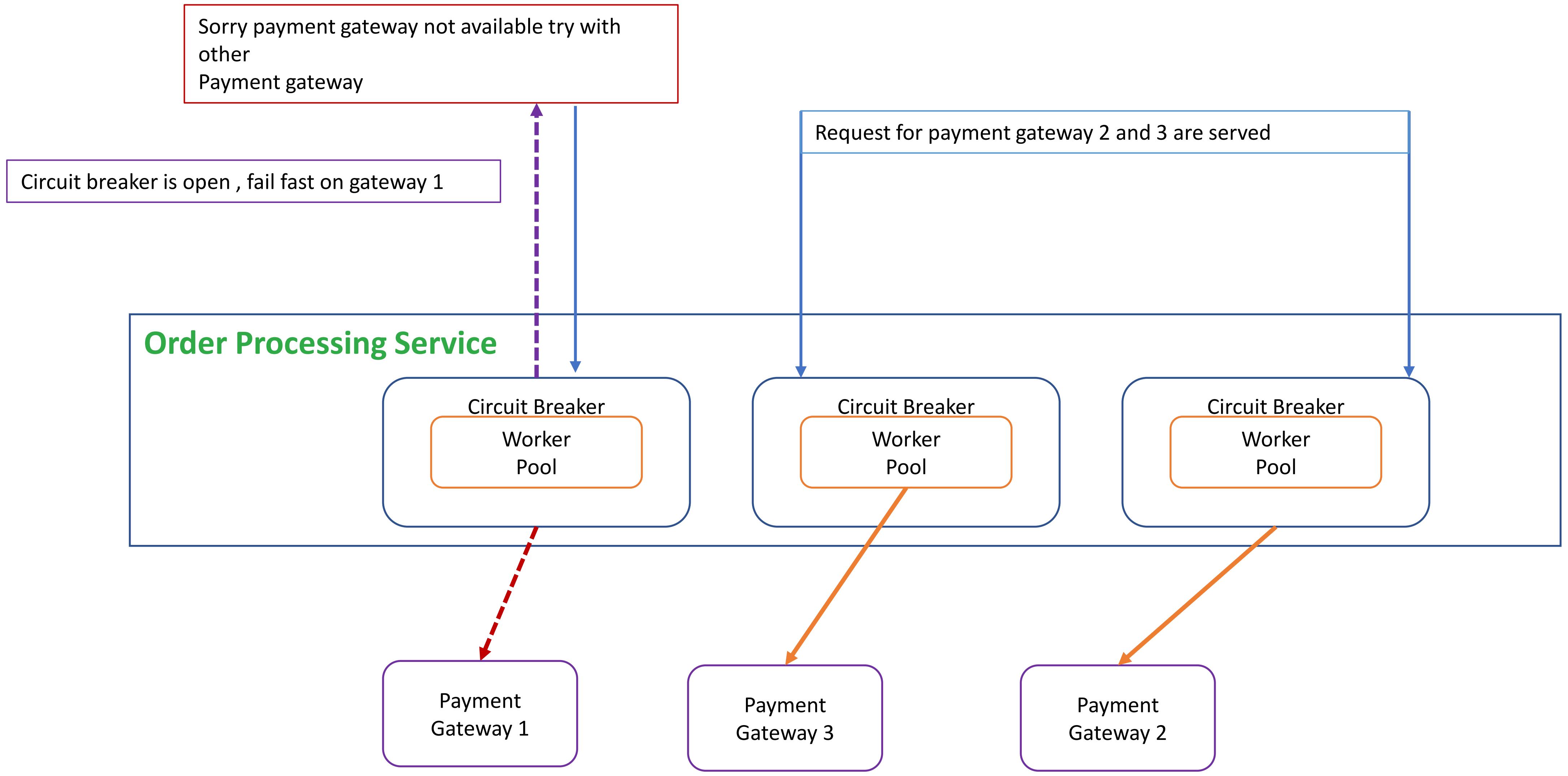
Ownership Culture

- Development
- Maintenance
- Business Team Ownership
- API Catalogue
- Architects

Resiliency

- Failure Is Everywhere
- How Much Is Too Much?
- Degrading Functionality
- Redundancy
- CAP Theorem
- Chaos Engineering

Resiliency : Circuit Breaker Pattern


Sorry payment gateway not available try with other Payment gateway
Circuit breaker is open , fail fast on gateway 1
Request for payment gateway 2 and 3 are served
Order Processing Service
Circuit Breaker
Worker Pool
Circuit Breaker
Worker Pool
Circuit Breaker
Worker Pool
Payment Gateway 1
Payment Gateway 3
Payment Gateway 2

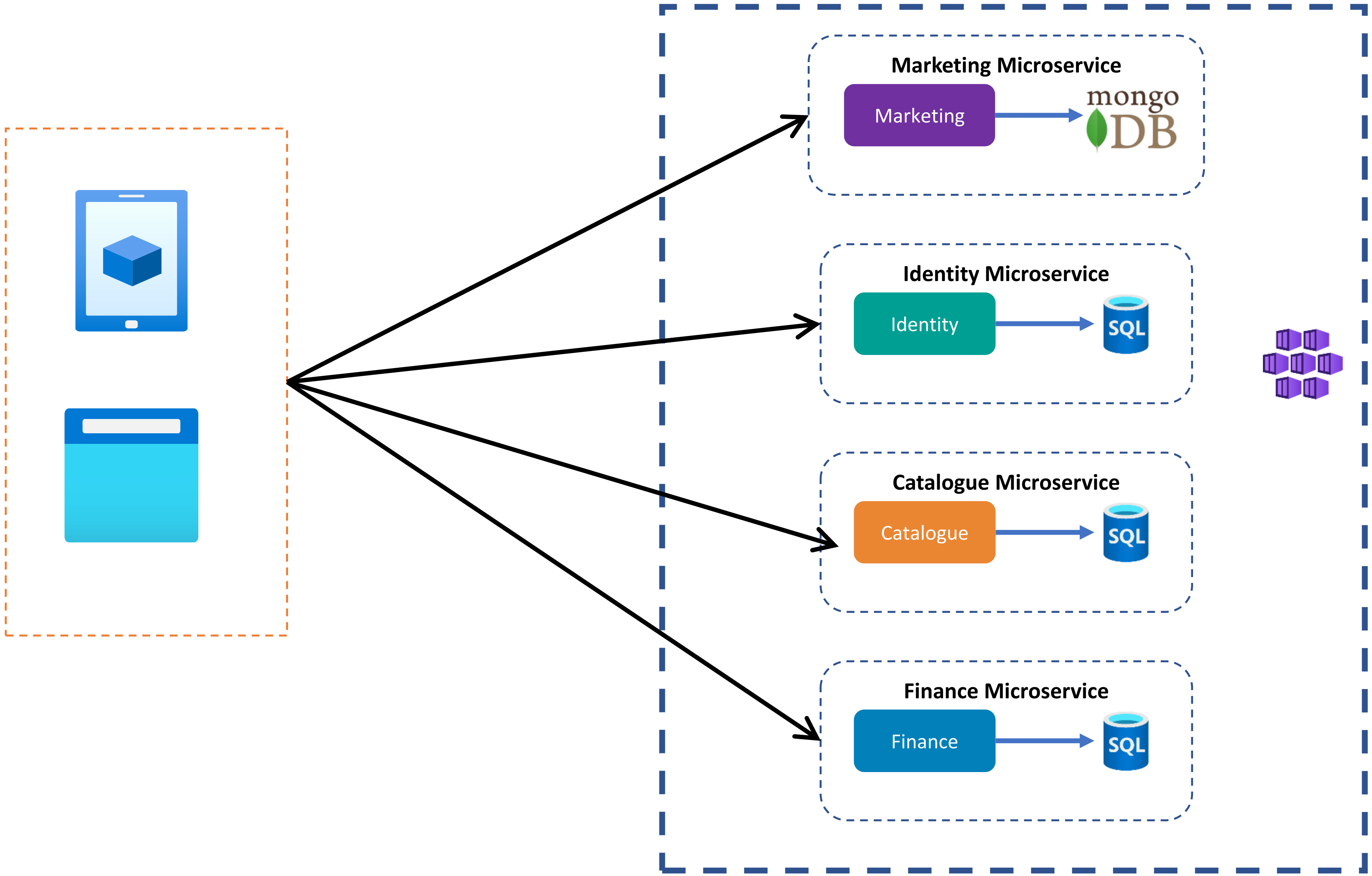
Resiliency Pattern

- Circuit Breaker Pattern
- Timeout Pattern
- Retry Pattern
- Caching Strategy
- Asynchronous Communication using Message Broker
- Active Backup
- Redundancy
- Maintain Network Health → Central Monitoring
- Data validation and error handling
- Centralized Security
- Chaos Engineering

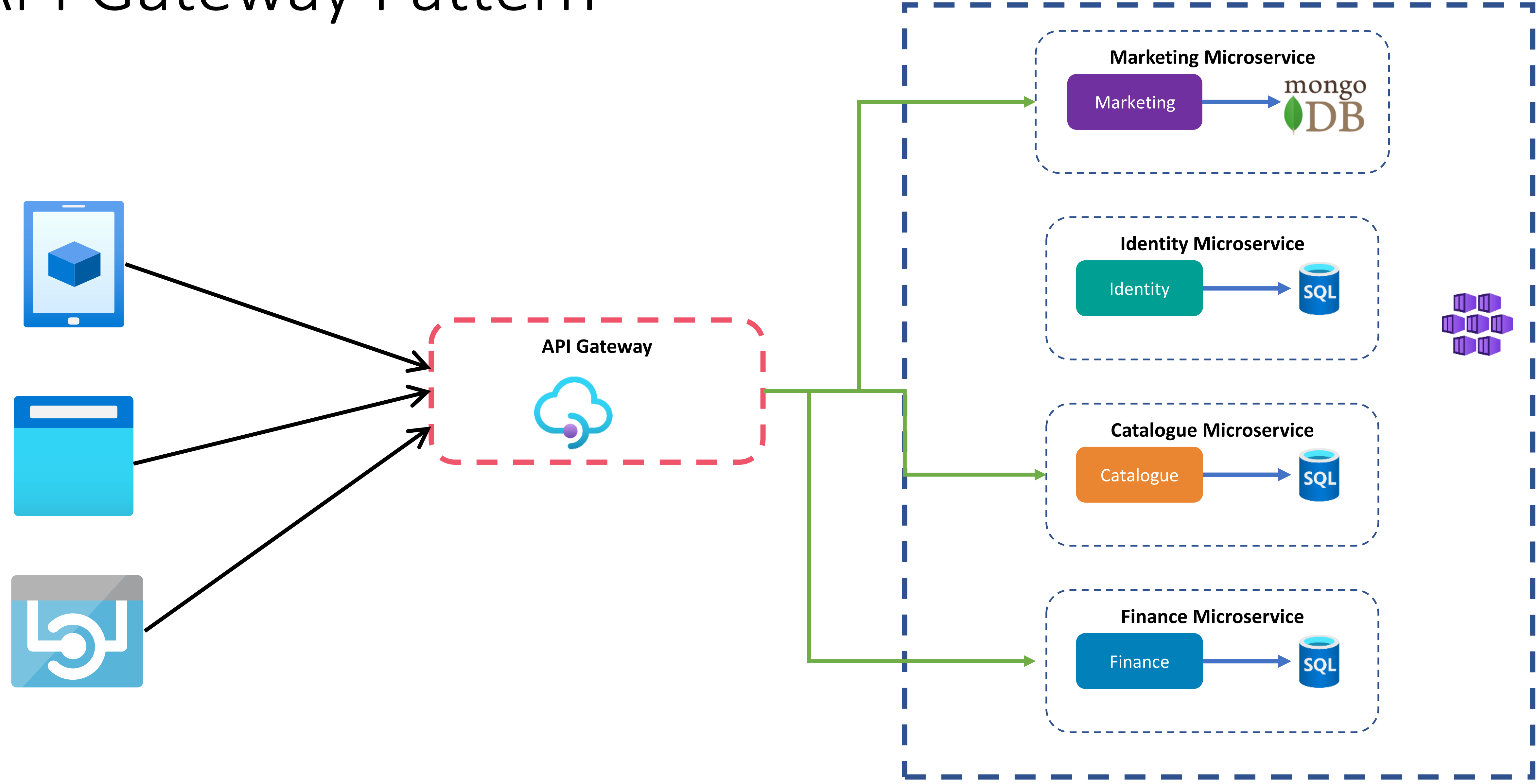
Communicating Between Microservices

- Front End to Micro Service Communication
- Micro Service to Micro Service Communication
- API Gateway
- Back End for Frontend (BFF)
- Synchronous Blocking Communication
- Asynchronous Communication
  - Communication through common data
  - Request/Response
  - Events

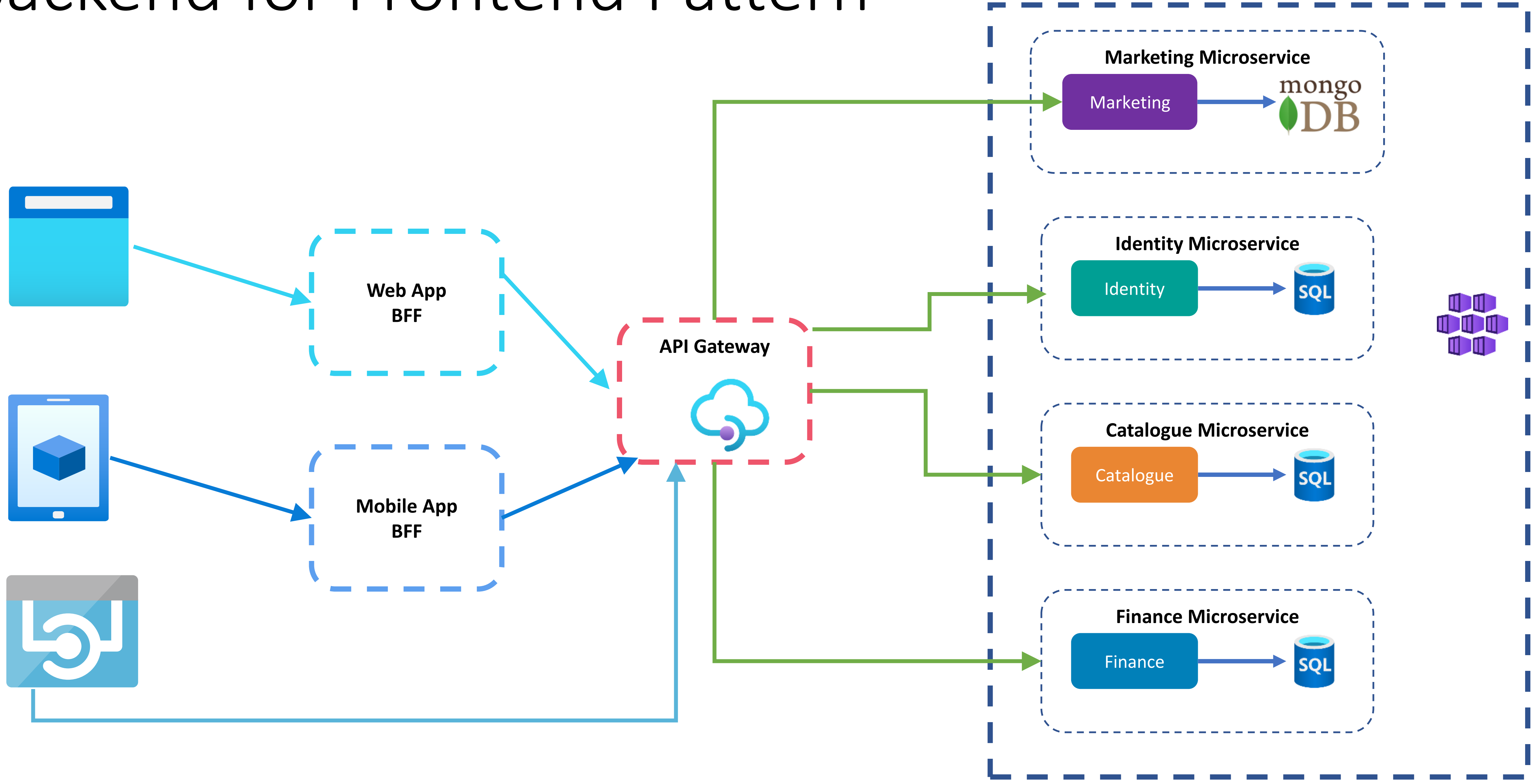
Direct Client To Service Communication



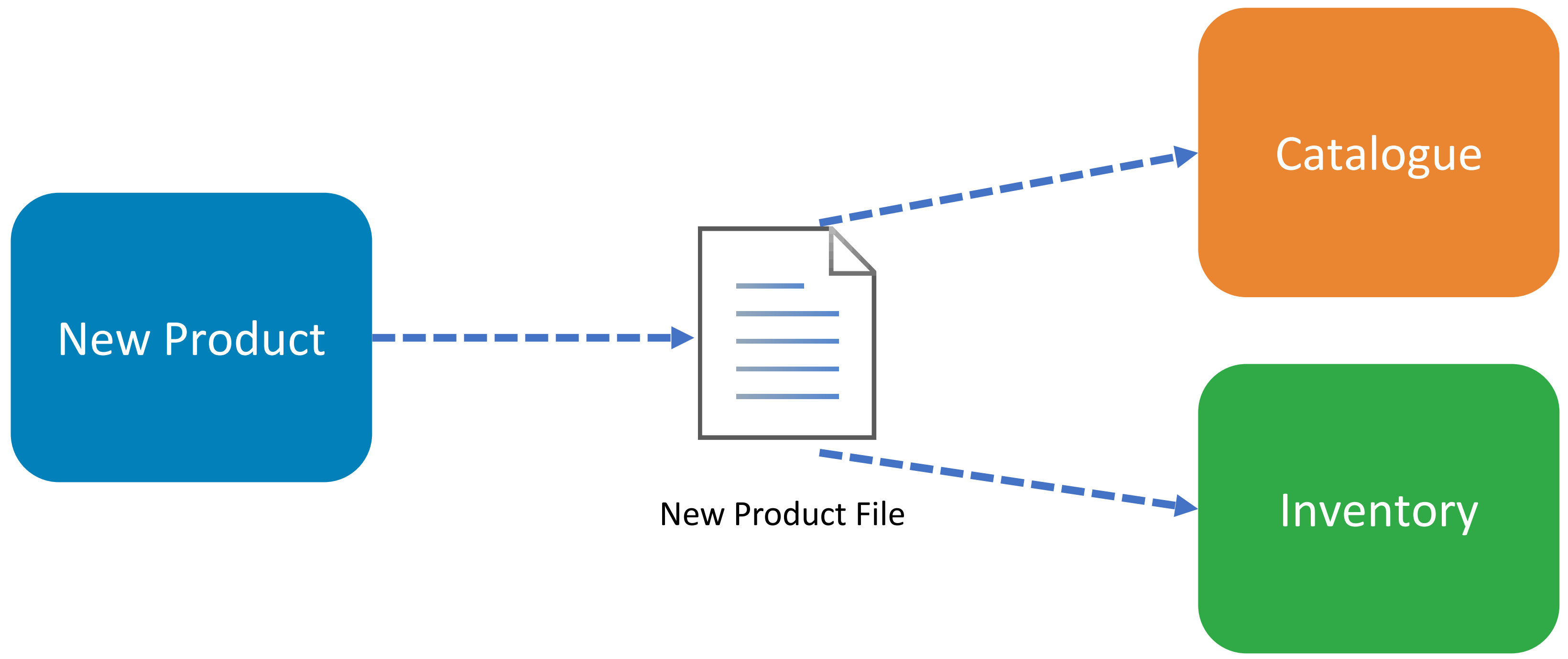
API Gateway Pattern



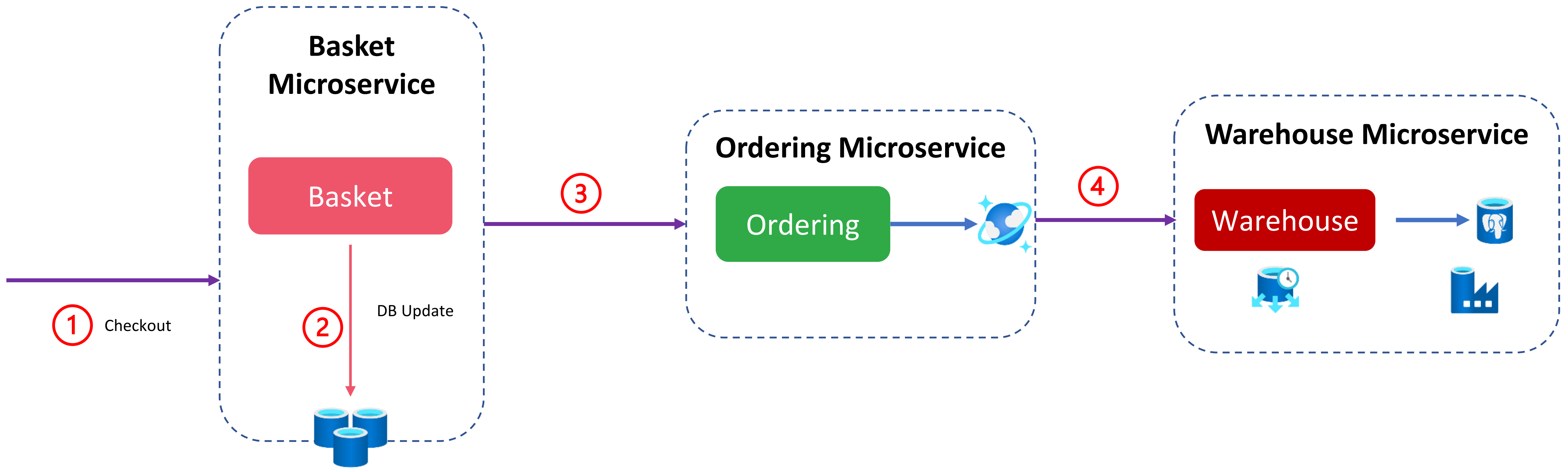
Backend for Frontend Pattern



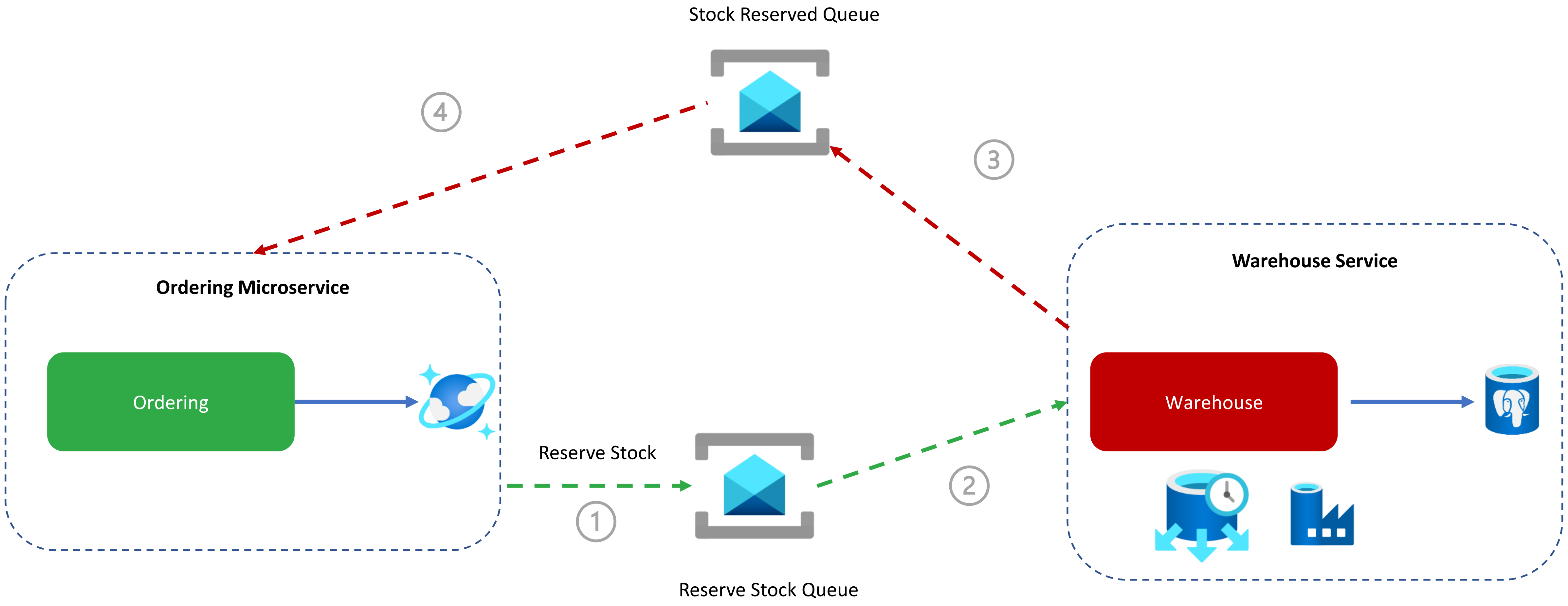
Communication Through Common Data



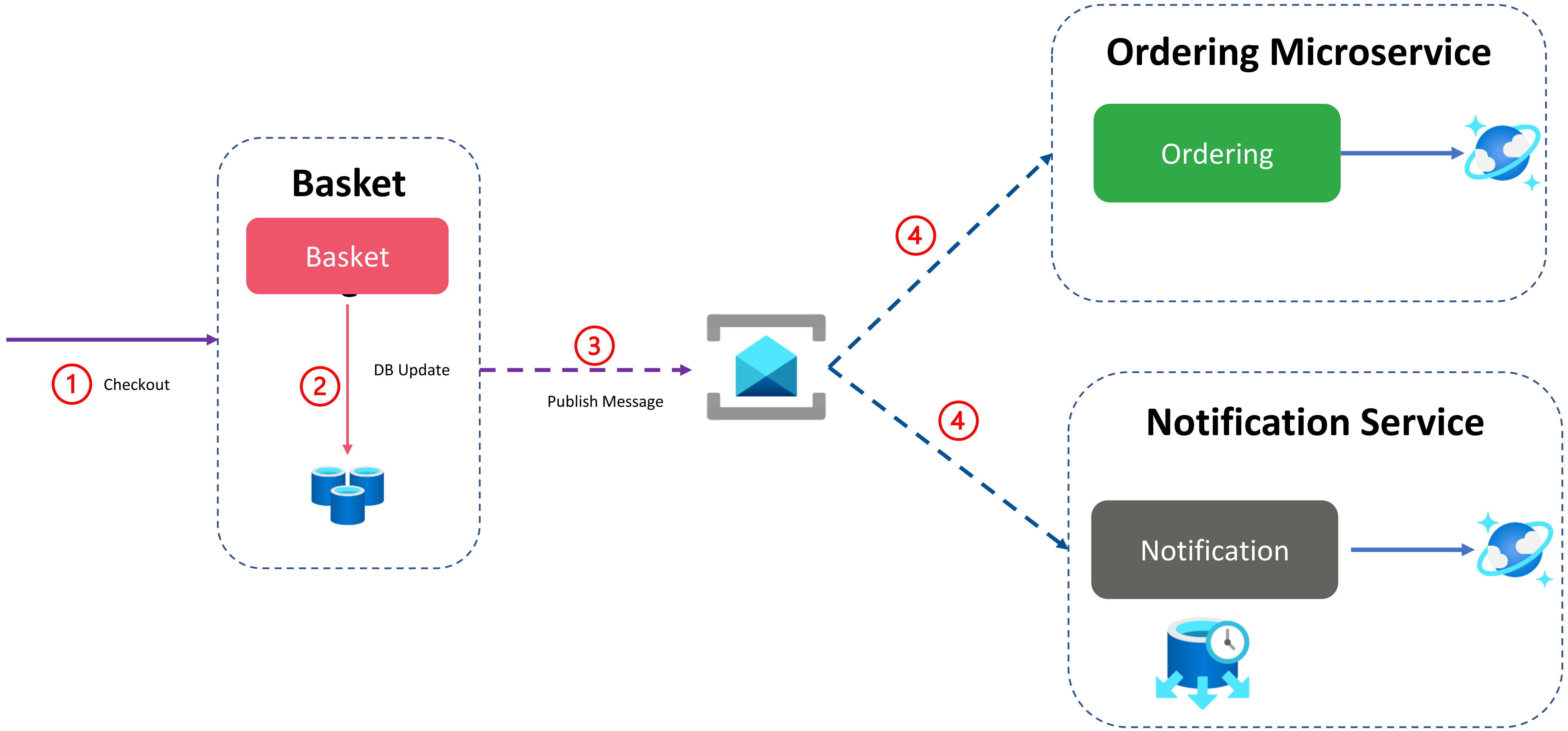
Request/Response : Synchronous Blocking



Request/Response : Asynchronous



Event-Driven Communication



Observability

- Log Aggregation
- Metrics Aggregation
- Distributed Tracing
- Site Reliability Engineering
    - Service Level Indicator
    - Service Level Objectives
    - Error Budget
- Alerting
- Chaos Engineering

SRE Books : https://sre.google/books/

Observability : Central Logging

- Scale Out
- Monitoring
- Alerts
- Log Archival
- Secure Logging
- Logging Library

- Azure Log Analytics
- Azure Application Insight

Automation : Microservices Tests

- Unit tests
- Contract tests
- Performance tests
- Security tests
- Integration tests : Test single microservices
- System tests : End to end scenarios.

Automation : Microservices Deployment

- Isolated Execution
- Focus on Automation
- Infrastructure as Code
- Zero Downtime Deployment
- State Management

Automation : Deployment Options

- Physical Machine
- Virtual Machine
- Container
- Platform as Service
- Serverless

Automation : Deployment VS Release

Deployment is a shift of software from one controlled environment to another. On the other hand, releases are a collection of changes for users to experience

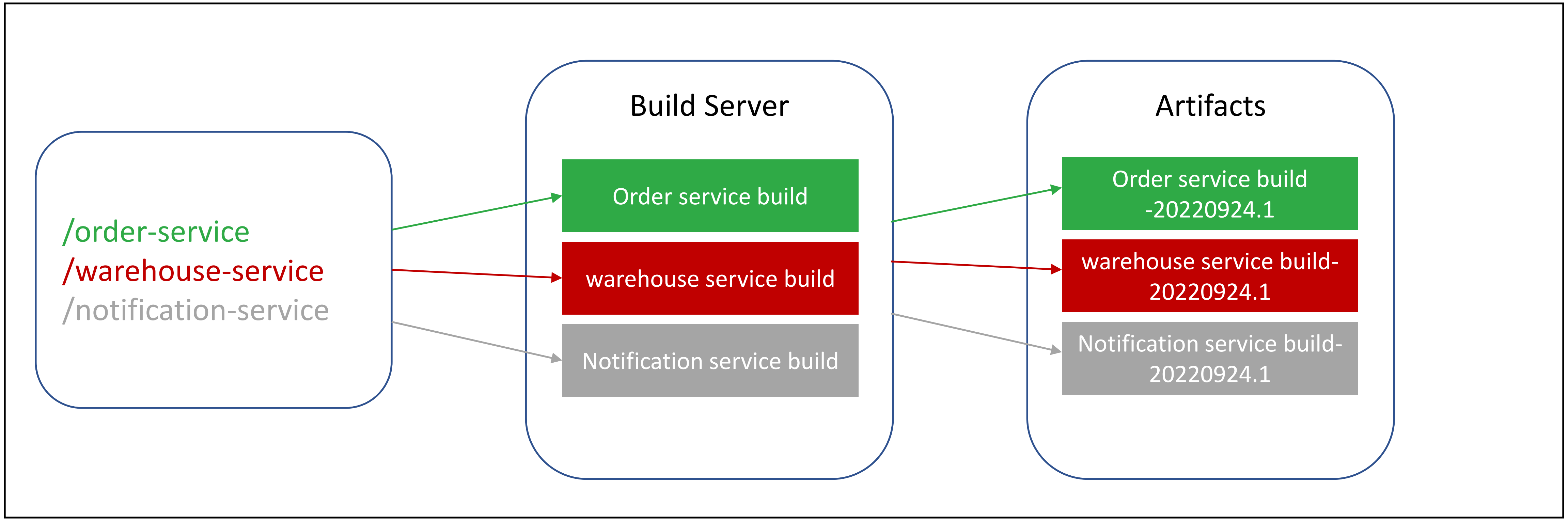
1. Feature Flag
2. Canary Release
3. Parallel Run

Automation : DevOps

- DevOps
  - Continuous Integration
  - Continuous Deployment
    - Automated
    - Semi Automated
  - Automatic builds
    - Daily builds
- Automated Tests
- Release Management
- Infrastructure as code

Microservice Source code Management

- One big repo→ one big deployment
- One repo→ one microservice
- Pattern → monorepro


/order-service
/warehouse-service
/notification-service
Build Server
Order service build
warehouse service build
Notification service build
Artifacts
Order service build -20220924.1
warehouse service build-20220924.1
Notification service build-20220924.1

Standardizing microservices

- Logs
- Health check
- Configuration
- Build scripts
- Deployment scripts
- Security
  - Authentication
  - Authorization

Security

- HTTPS
- Firewall
- API Gateway
- Penetration Testing
- Security Qualification Process
- Central Key Management tool
- Central Security Component

- Azure KeyVault
- HashiCorp Valut

- OAuth2
- OpenId Connect
- Azure AD
- Identity Server

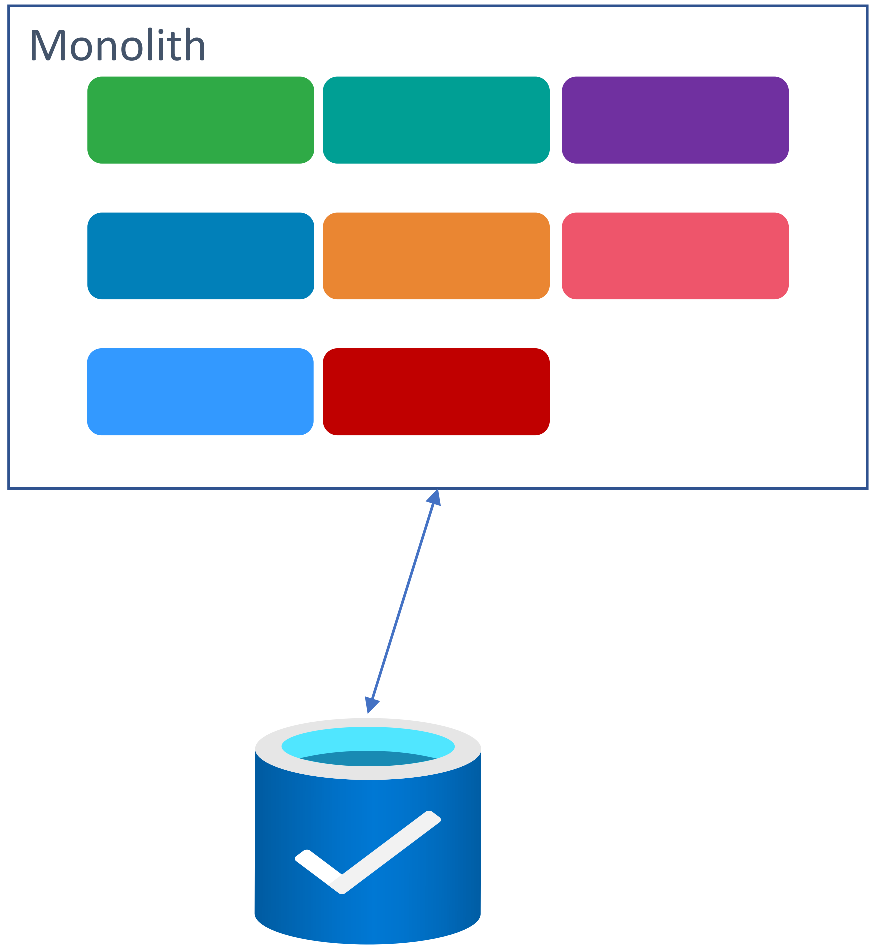
Monolith to Microservices

- Have a goal
- Identify other patterns to solve your problem
- Microservices is not the solution
- Data Decomposition Consideration
- Data Integrity
- Reporting

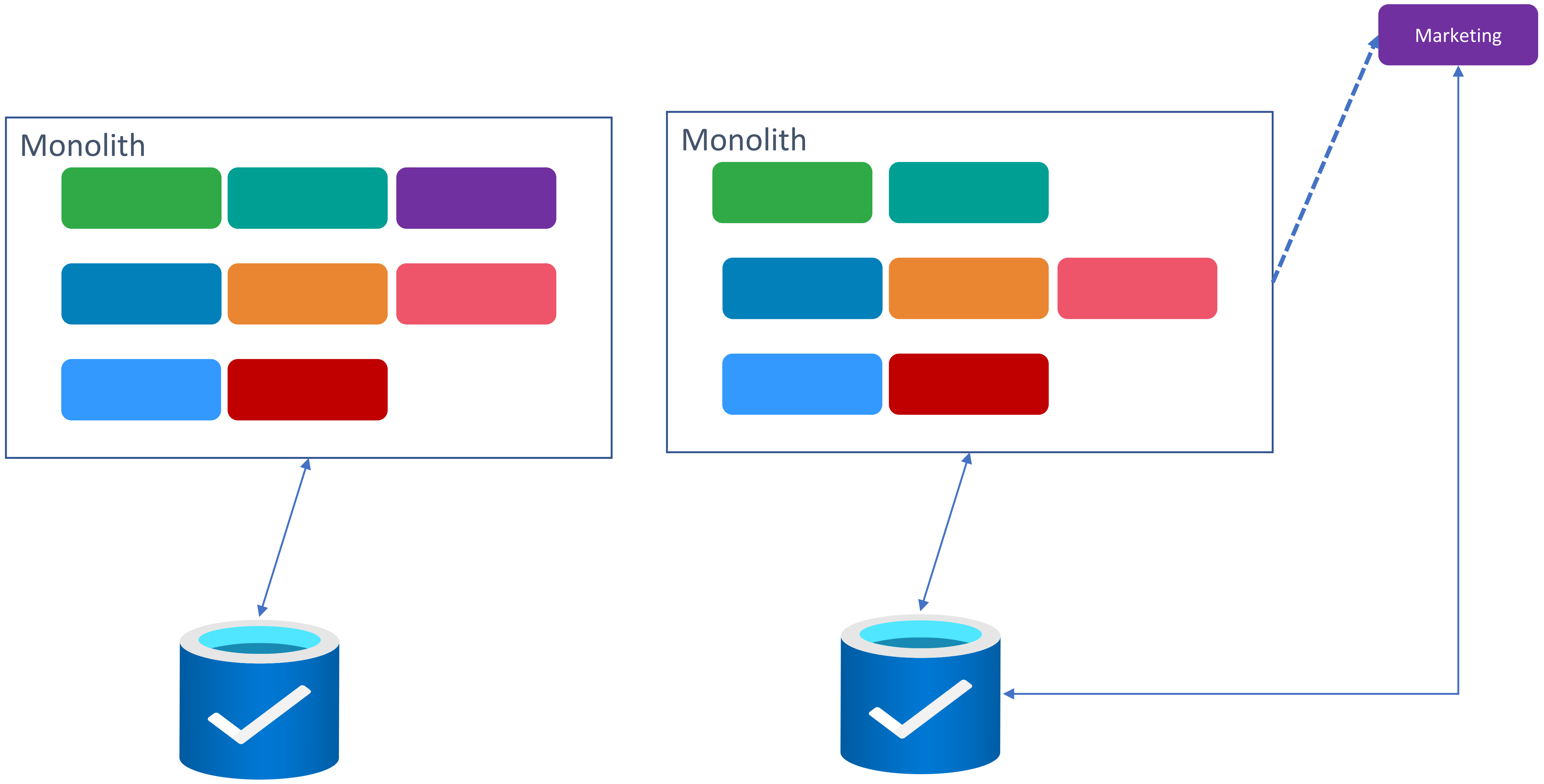
Monolith to Microservices

- Need a systematic approach
- Avoid big releases ("If you have big-bang rewrite, the only thing you are guaranteed of is a big bang" by Martin Fowler
- Start with low risk microservices
- Learning and deploying new microservices
- Strangler application pattern
  - Monolith Decomposition Patterns : https://bit.ly/MonolithDecomposition

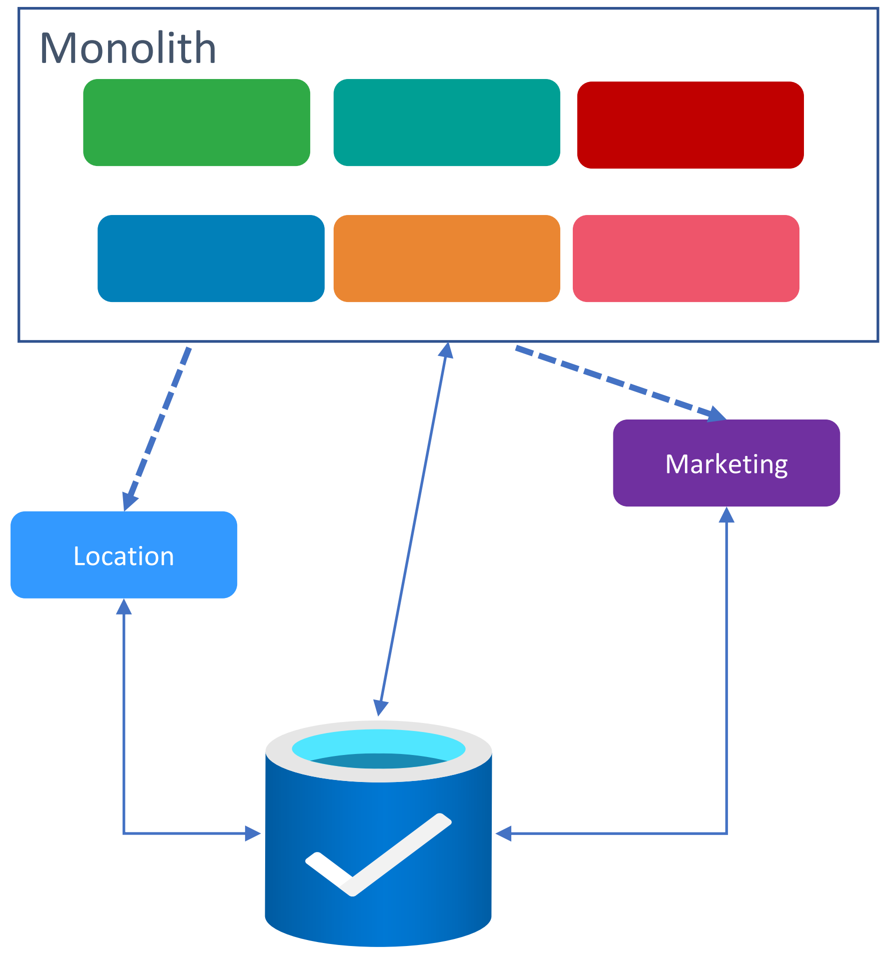
What to Split First

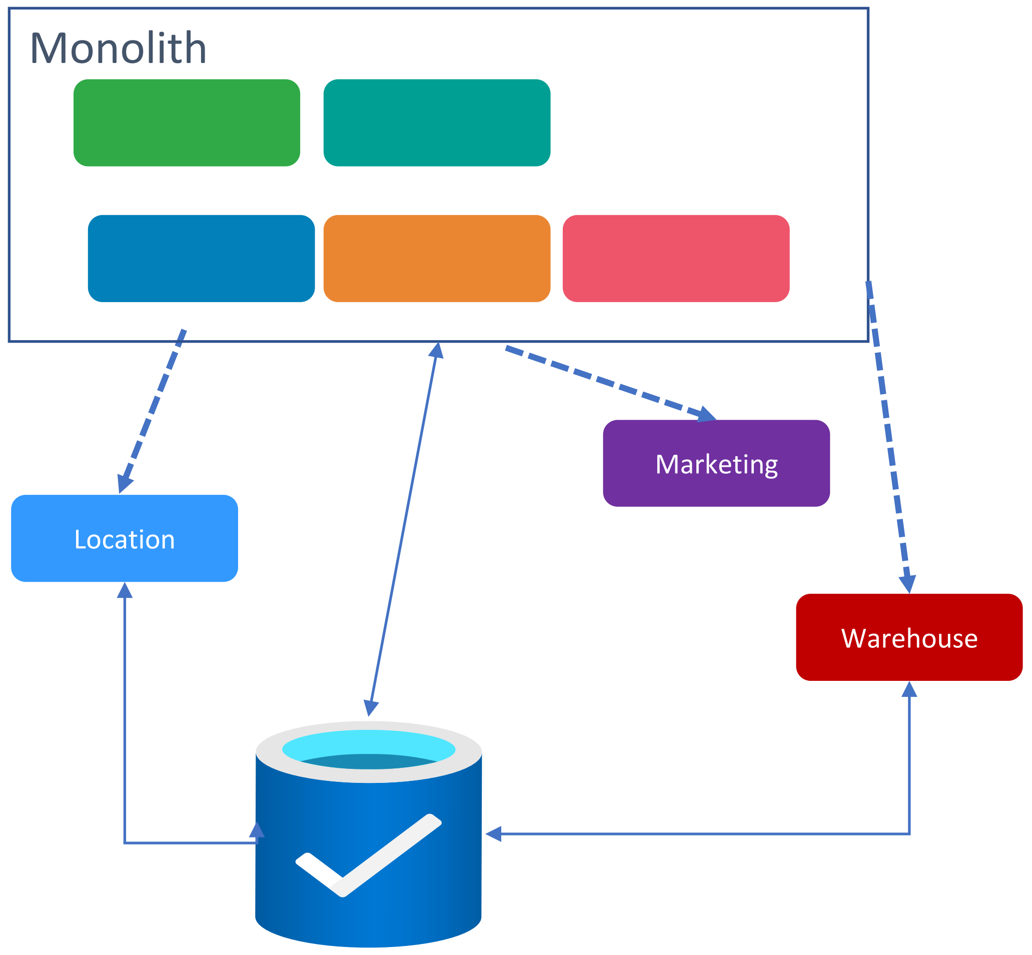


What to Split First - Code

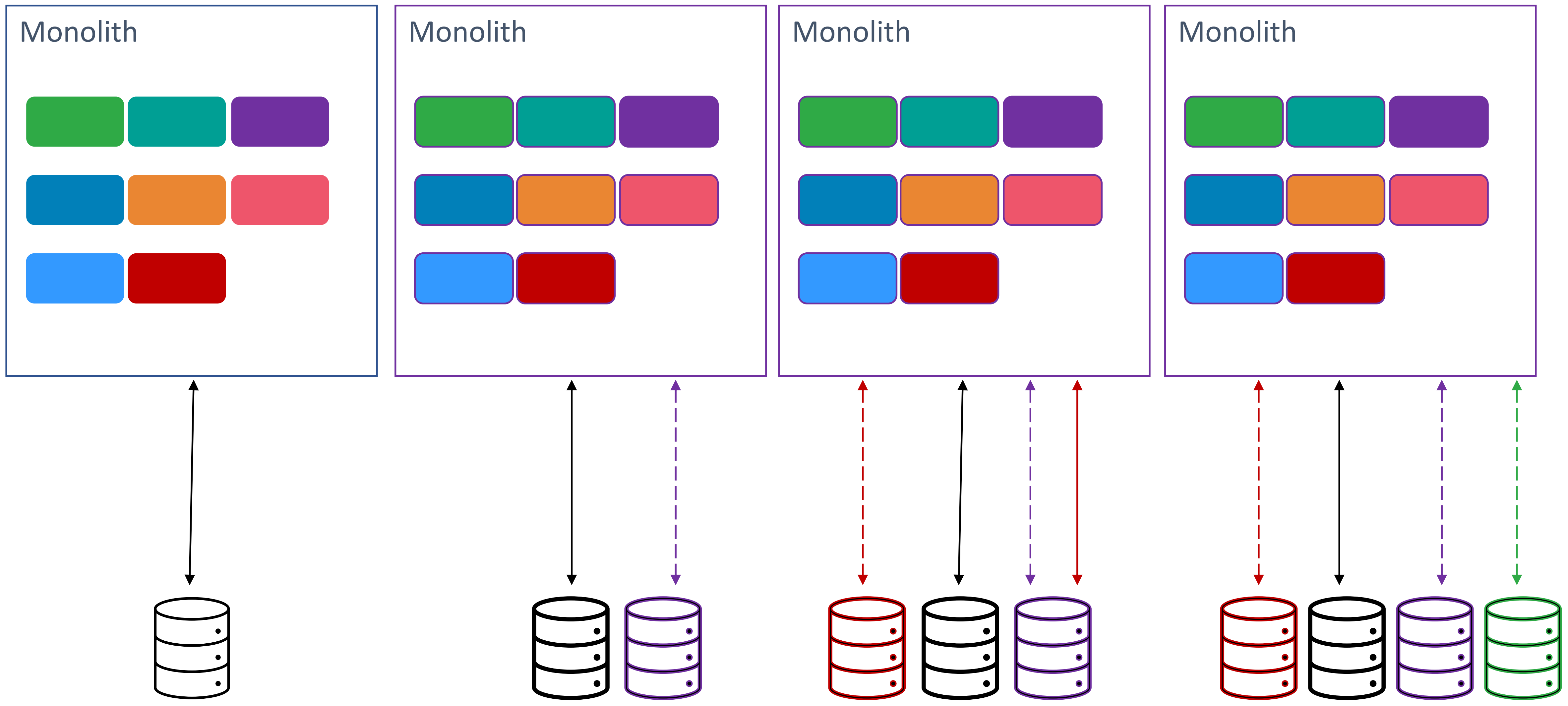


What to Split First - Code





What to Split First - Database



What to split first

- Parallel Run
- Feature Flag

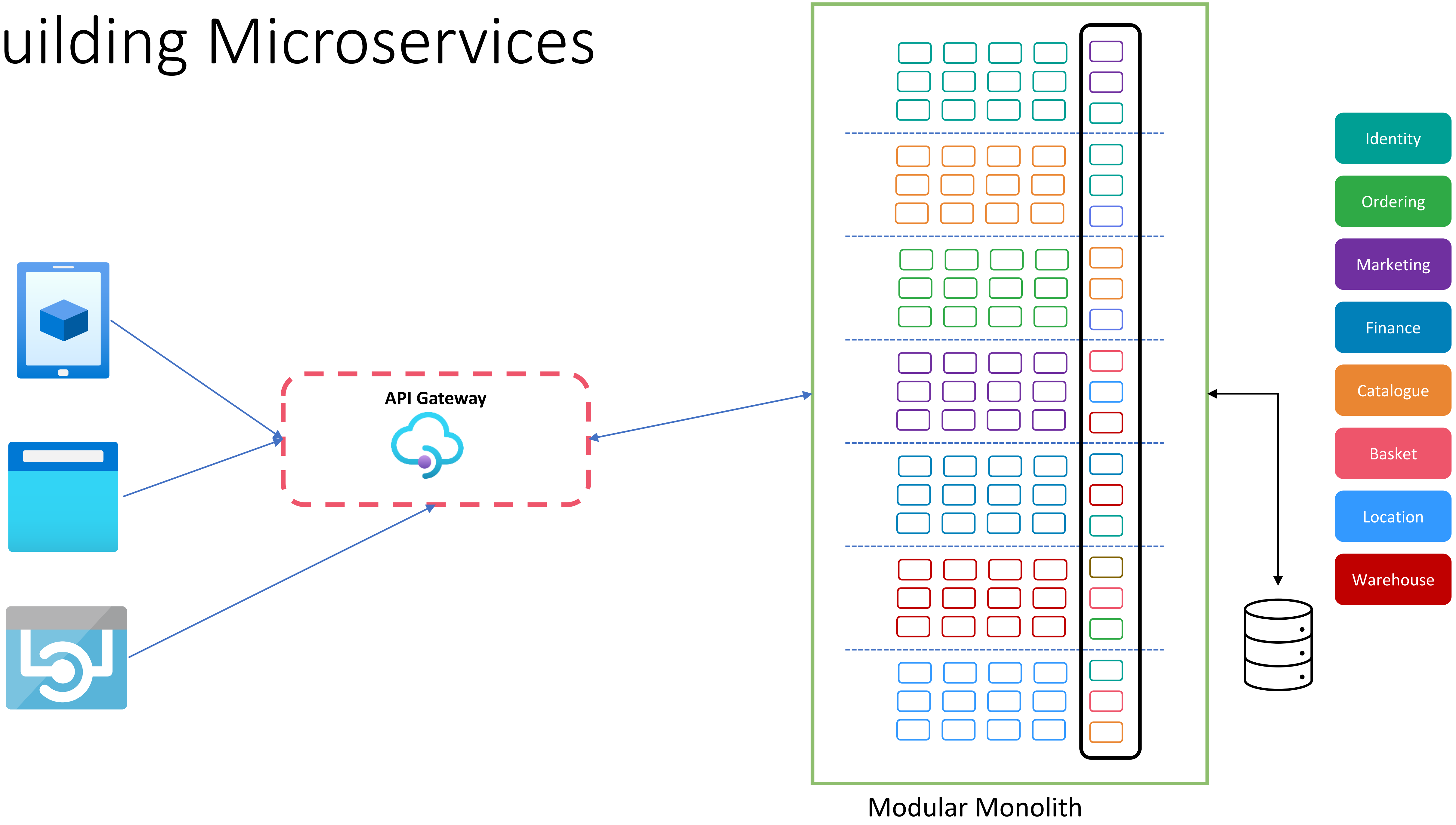
Building Microservices

- Automated deployments
- Automated tests
- Domain Driven design
- Split modular Monolith to microservices

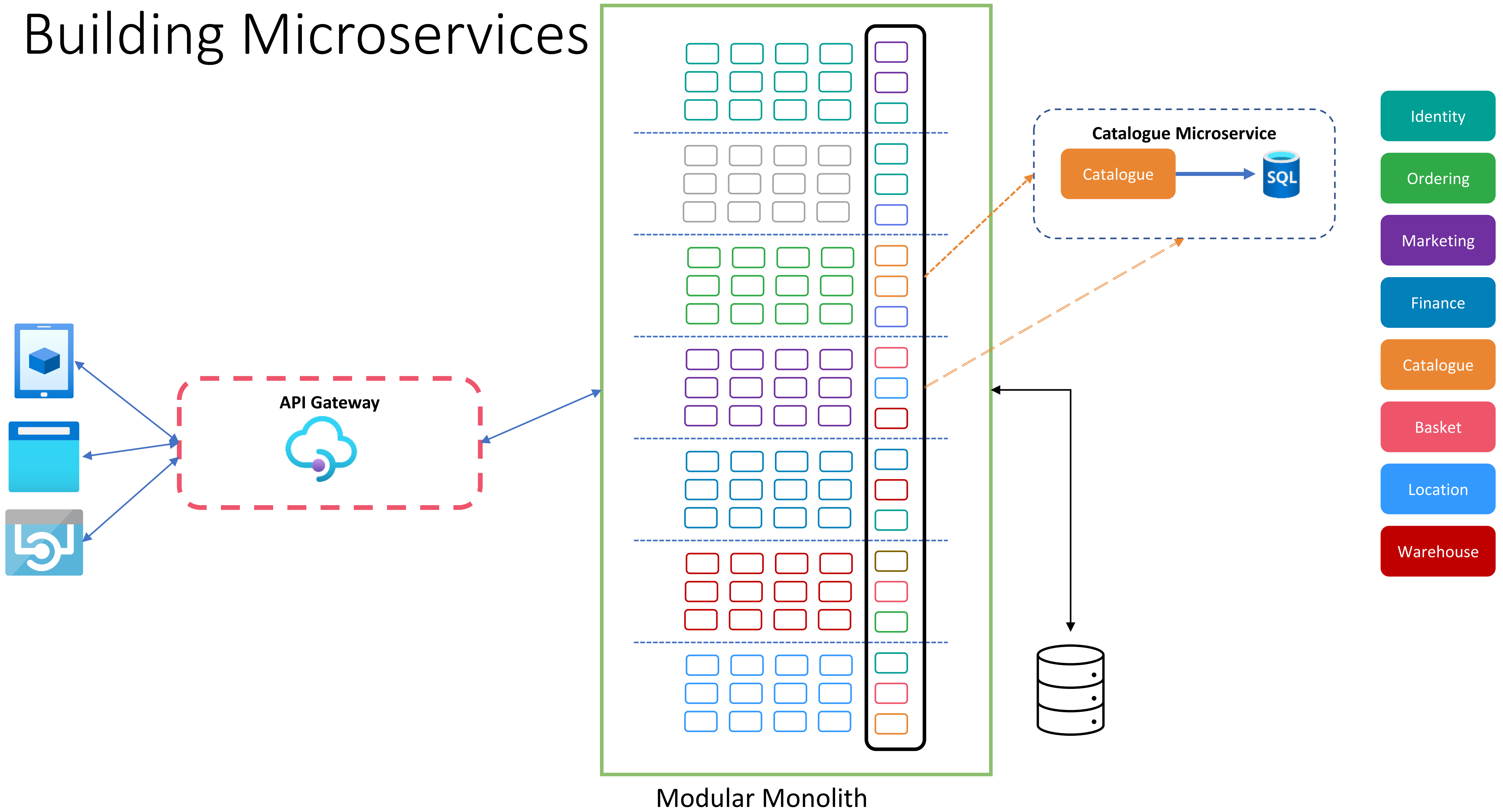
Building Microservices

- Limited Resources Teams
  - Start with Modular Monolith
  - Scale individual Modules
- Resourced Teams
  - Avoid Big Bang
  - Start Small
  - API Gateway
  - API Catalogue

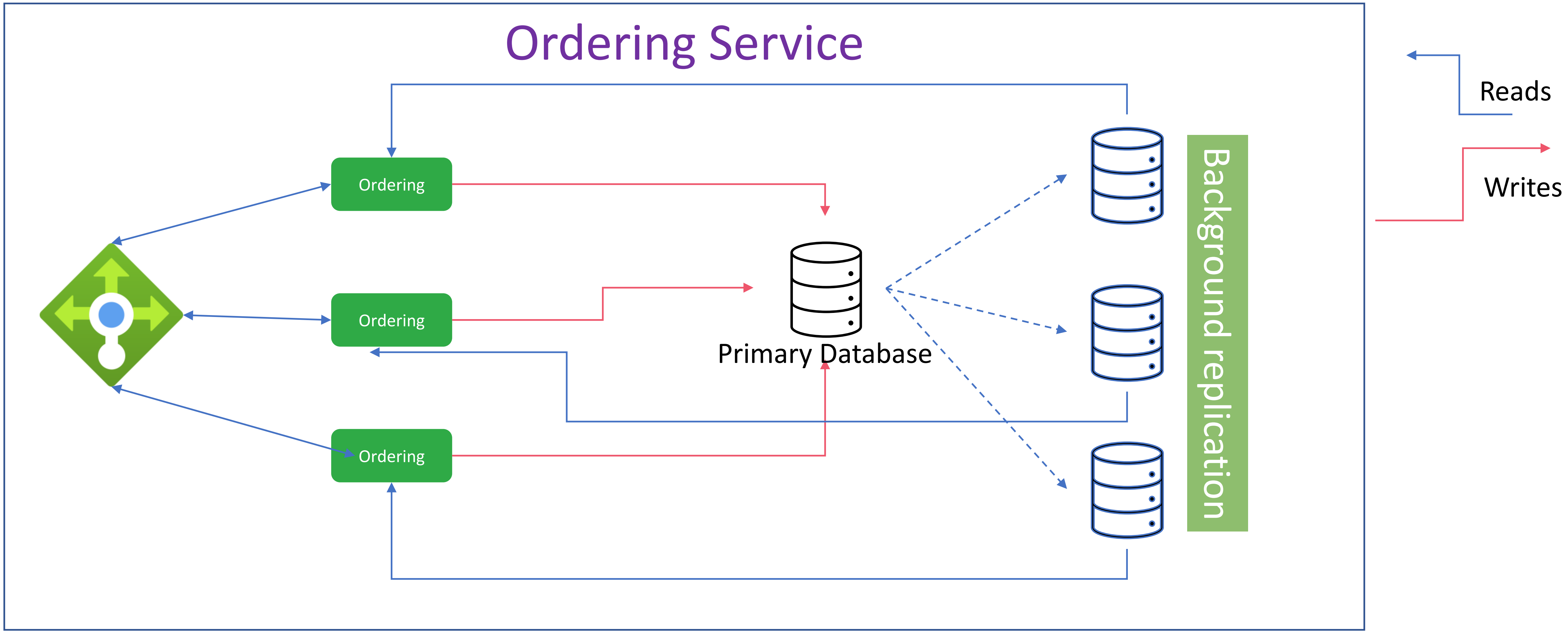
Building Microservices



Building Microservices



Database Deployment and Scaling



Data Integrity

- Transactions
  - I recommend the presentation “Google Cloud Spanner: Global Consistency at Scale : https://bit.ly/GoogleCloudSpanner
- Consistency

Summary

Summary

Autonomous

Services are independently deployable and changeable

Domain Driven Design

Services represent a specific business domain with a cohesive focus

Ownership Culture

Tret each service as a product, own it

Resiliency

Failure us everywhere. Design for failure

Observability

Health of service is visible and traceable.

Automation

Automation is key from check-in→ build→ test→ deploy and environment creation.

Q & A