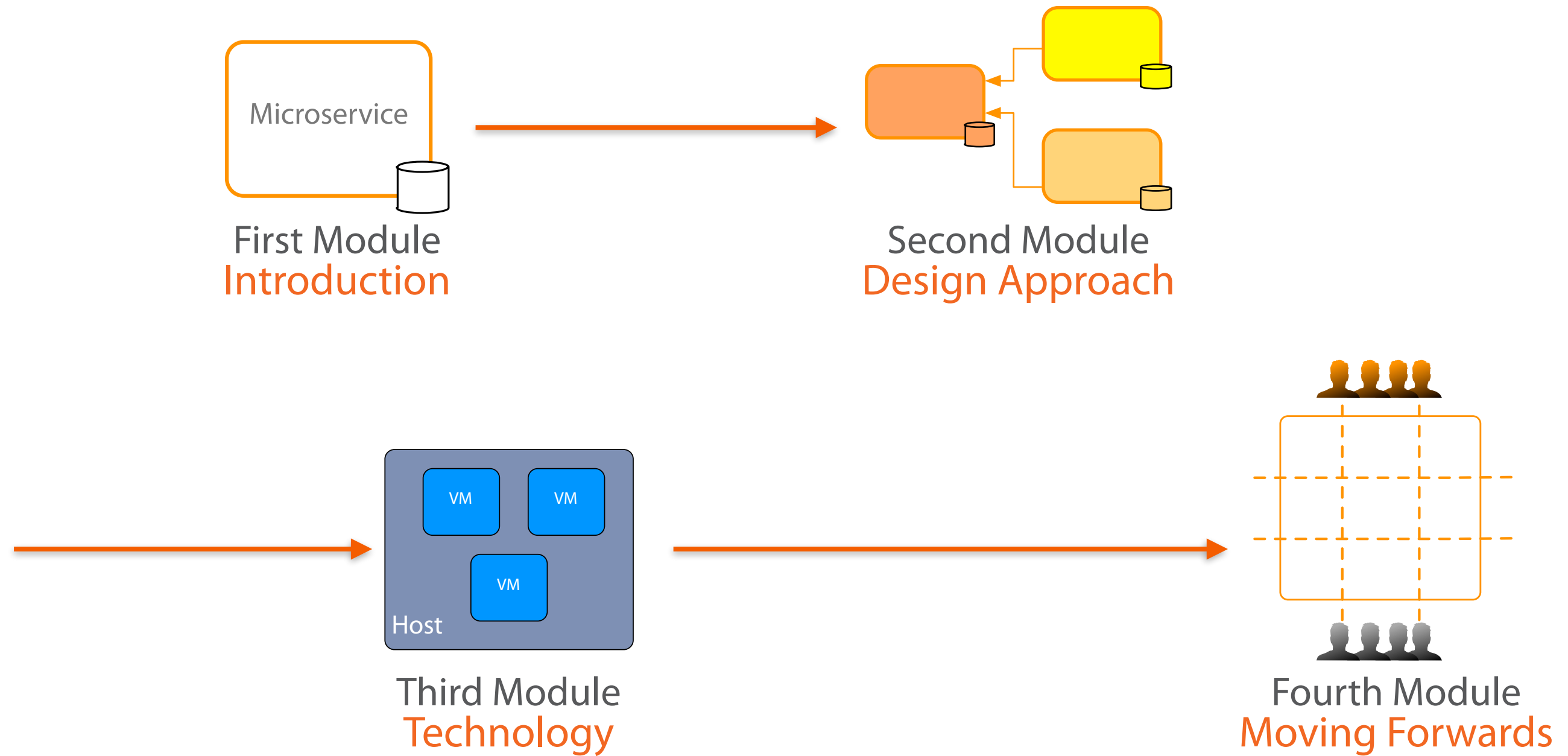
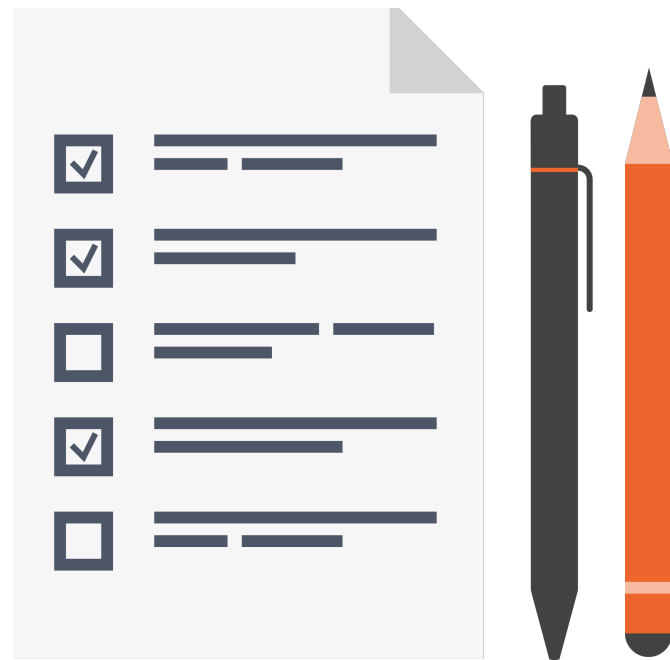


Course Overview



Module Overview



Microservices

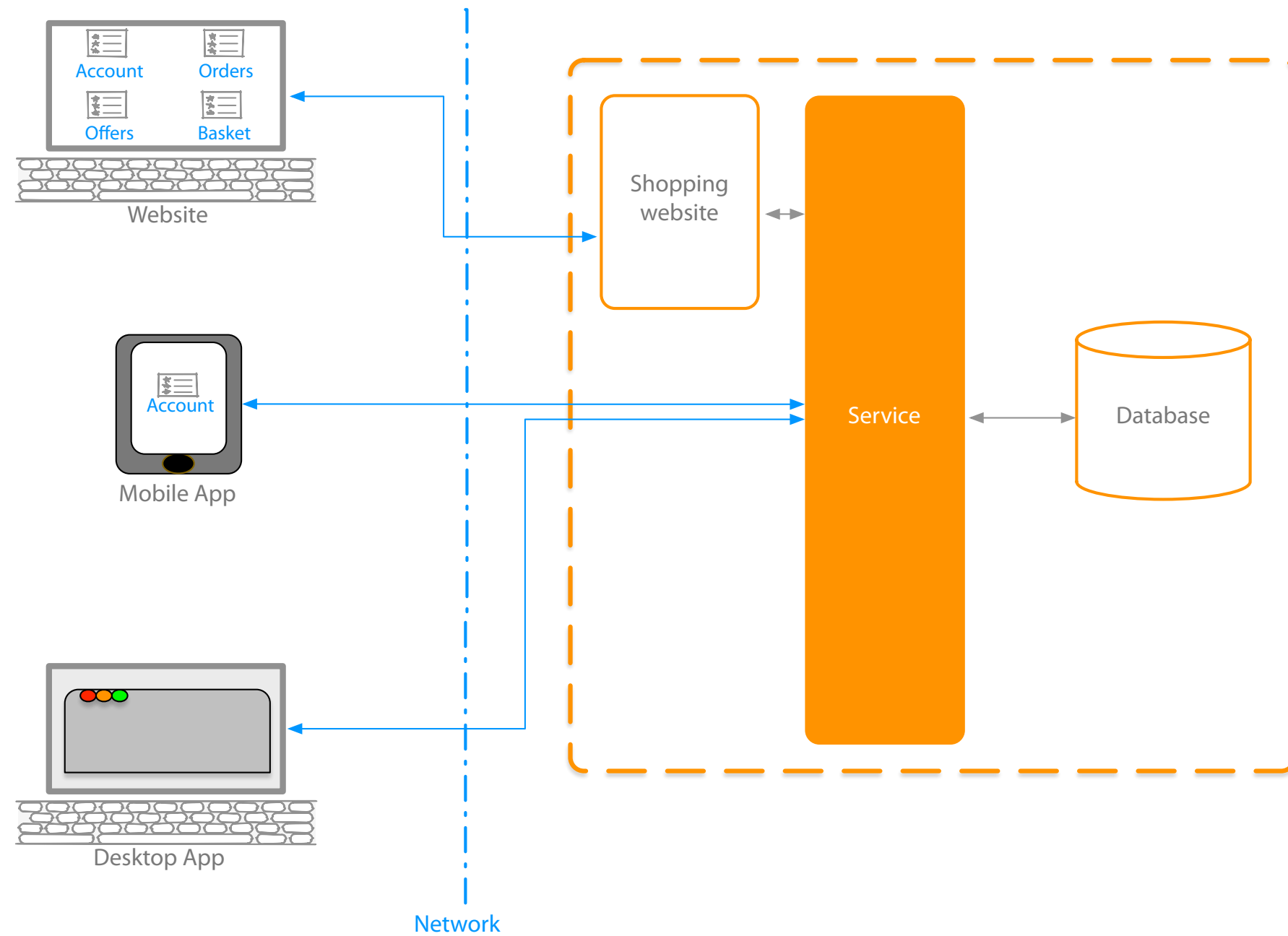
Emergence of Microservices

Microservices Design Principles

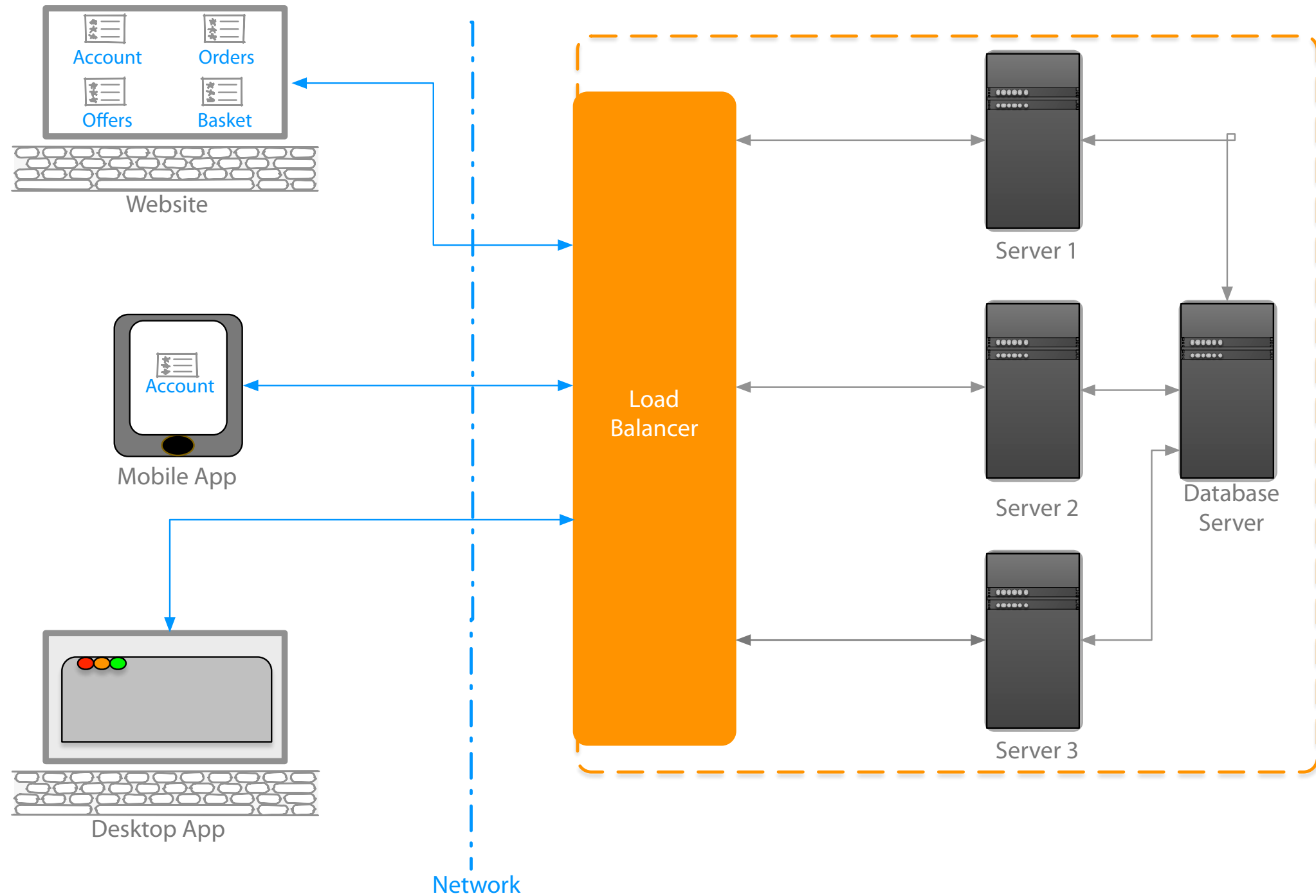
Microservices

What is a Service? | Introduction | The Monolithic

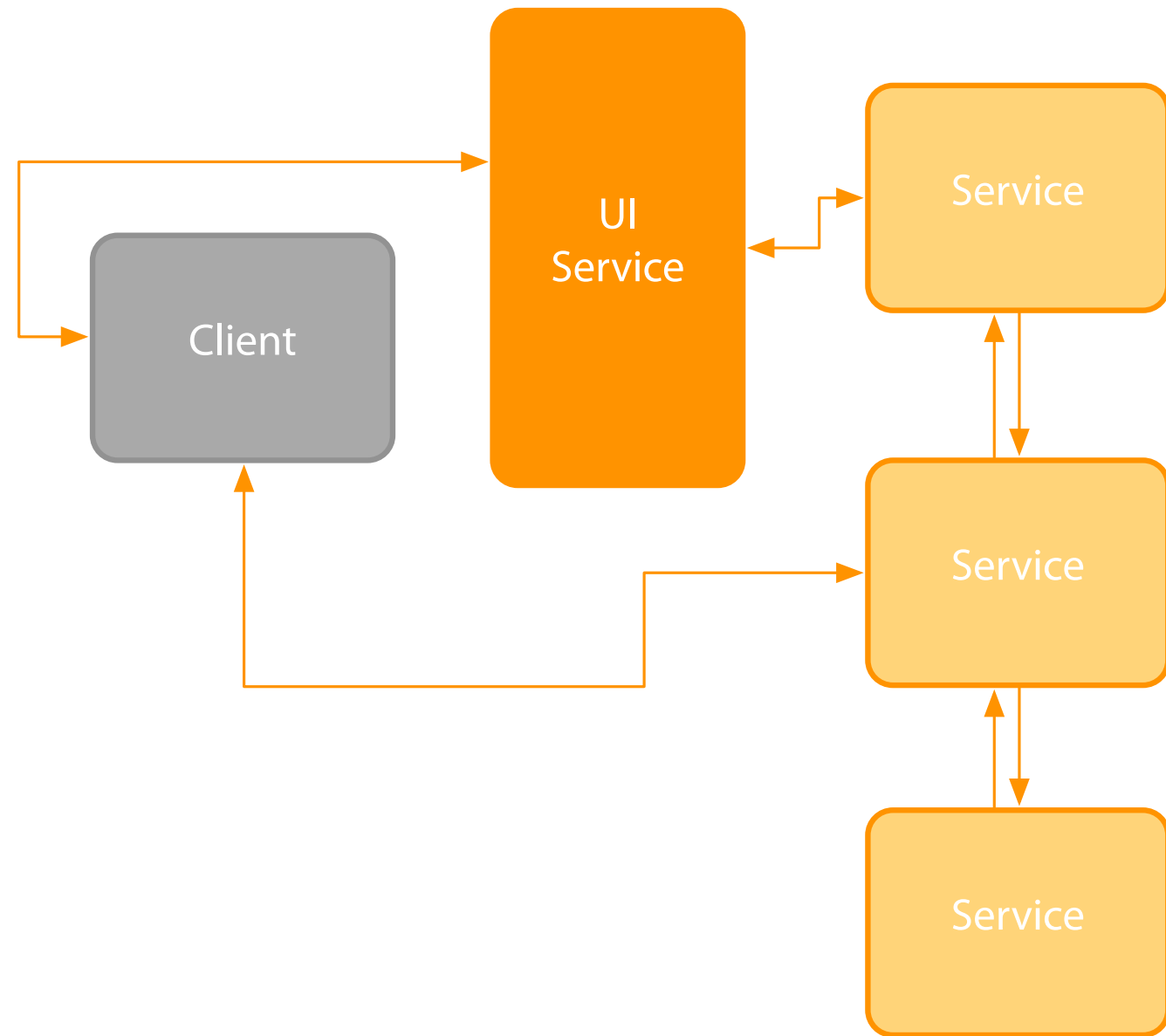
Microservices: What is a Service?



Microservices: What is a Service?



Microservices: Introduction



SOA done well

Knowing how to size a service

Traditional SOA resulted in monolithic services

Micro sized services provide

Efficiently scalable applications

Flexible applications

High performance applications

Application(s) powered by multiple services

Small service with a single focus

Lightweight communication mechanism

Both client to service and service to service

Technology agnostic API

Independent data storage

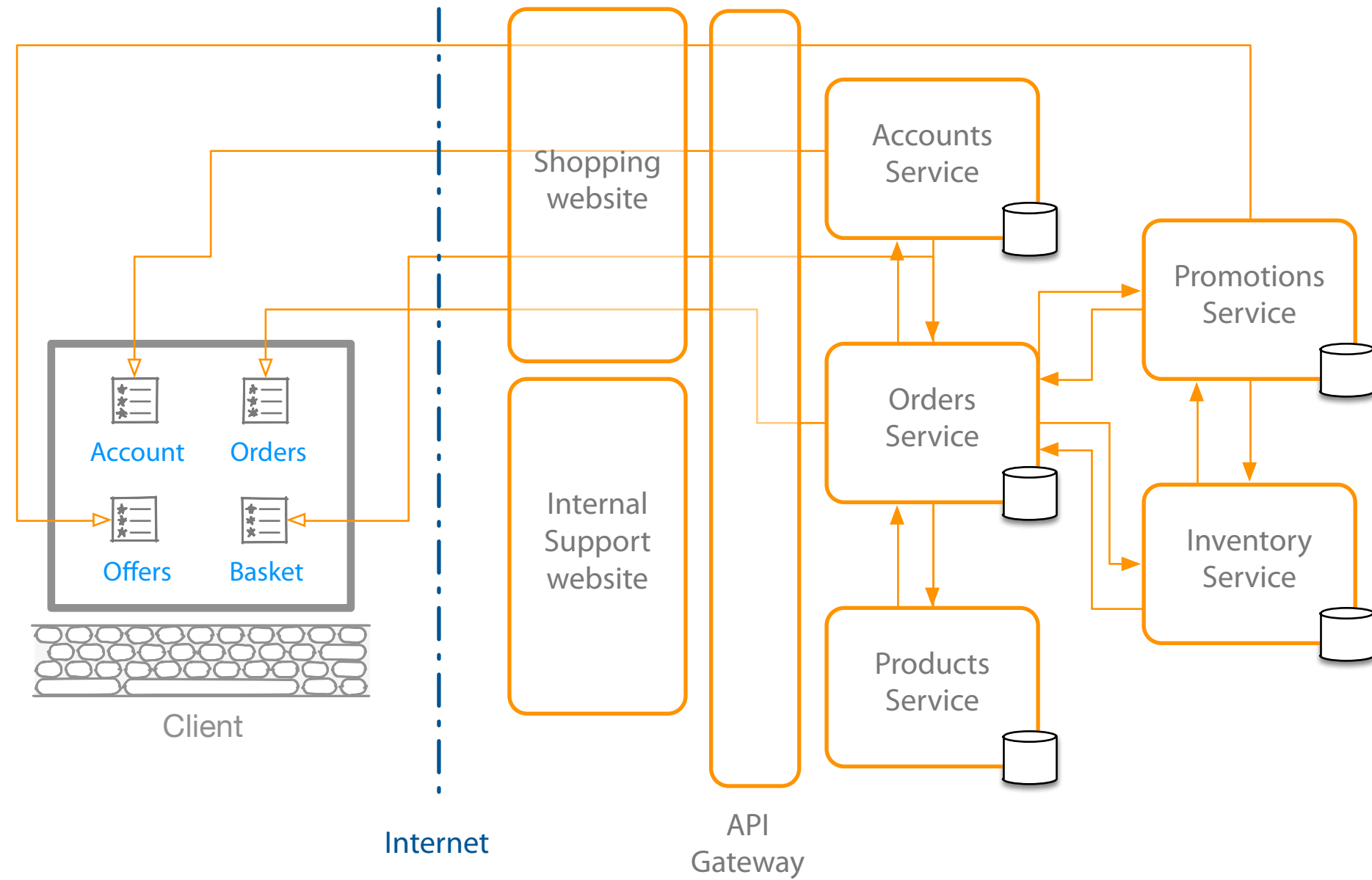
Independently changeable

Independently deployable

Distributed transactions

Centralized tooling for management

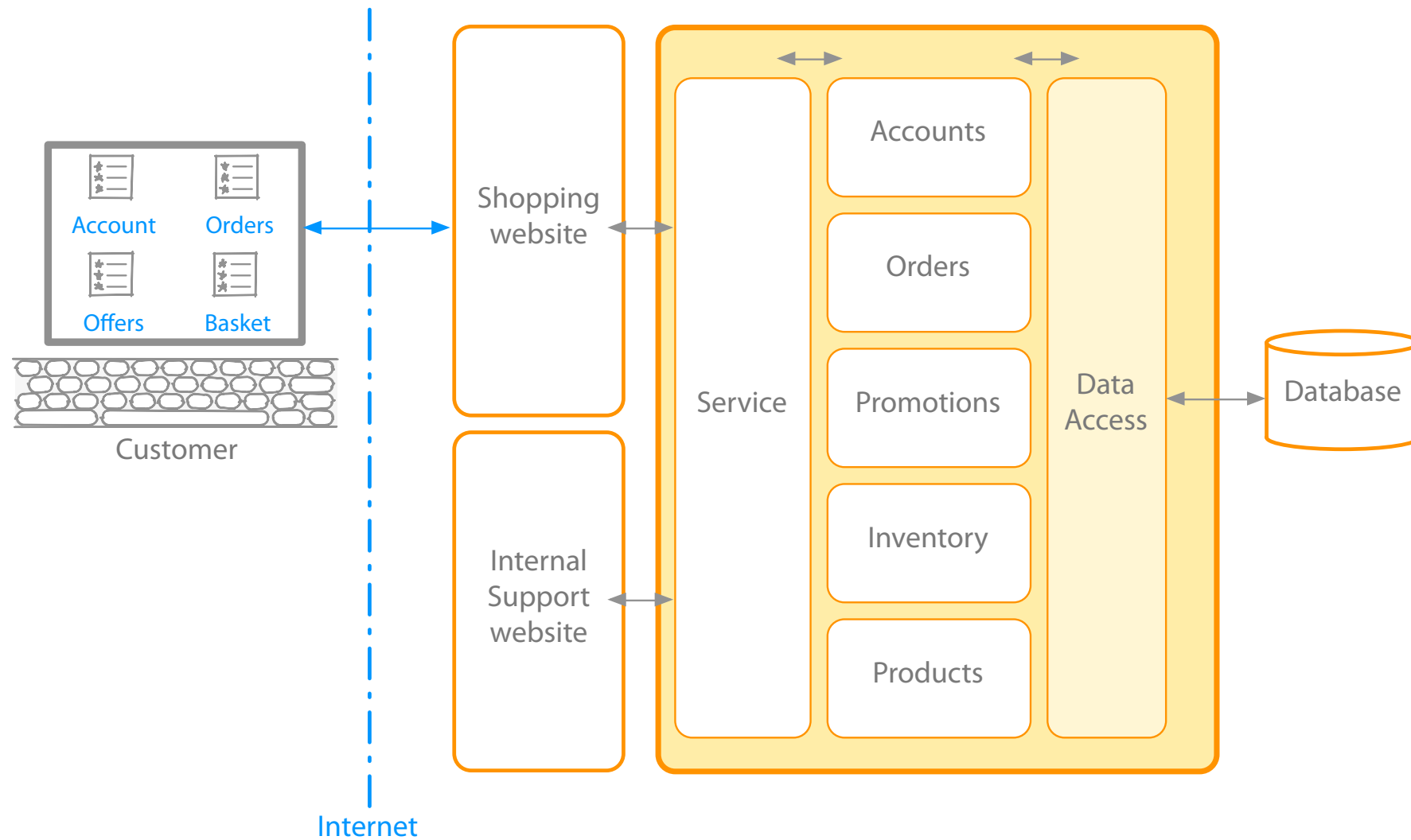
Microservices: Introduction



Microservices

What is a Service? | Introduction | The Monolithic

Microservices: The Monolithic



Typical enterprise application

No restriction on size

Large codebase

Longer development times

Challenging deployment

Inaccessible features

Fixed technology stack

High levels of coupling

Between modules

Between services

Failure could affect whole system

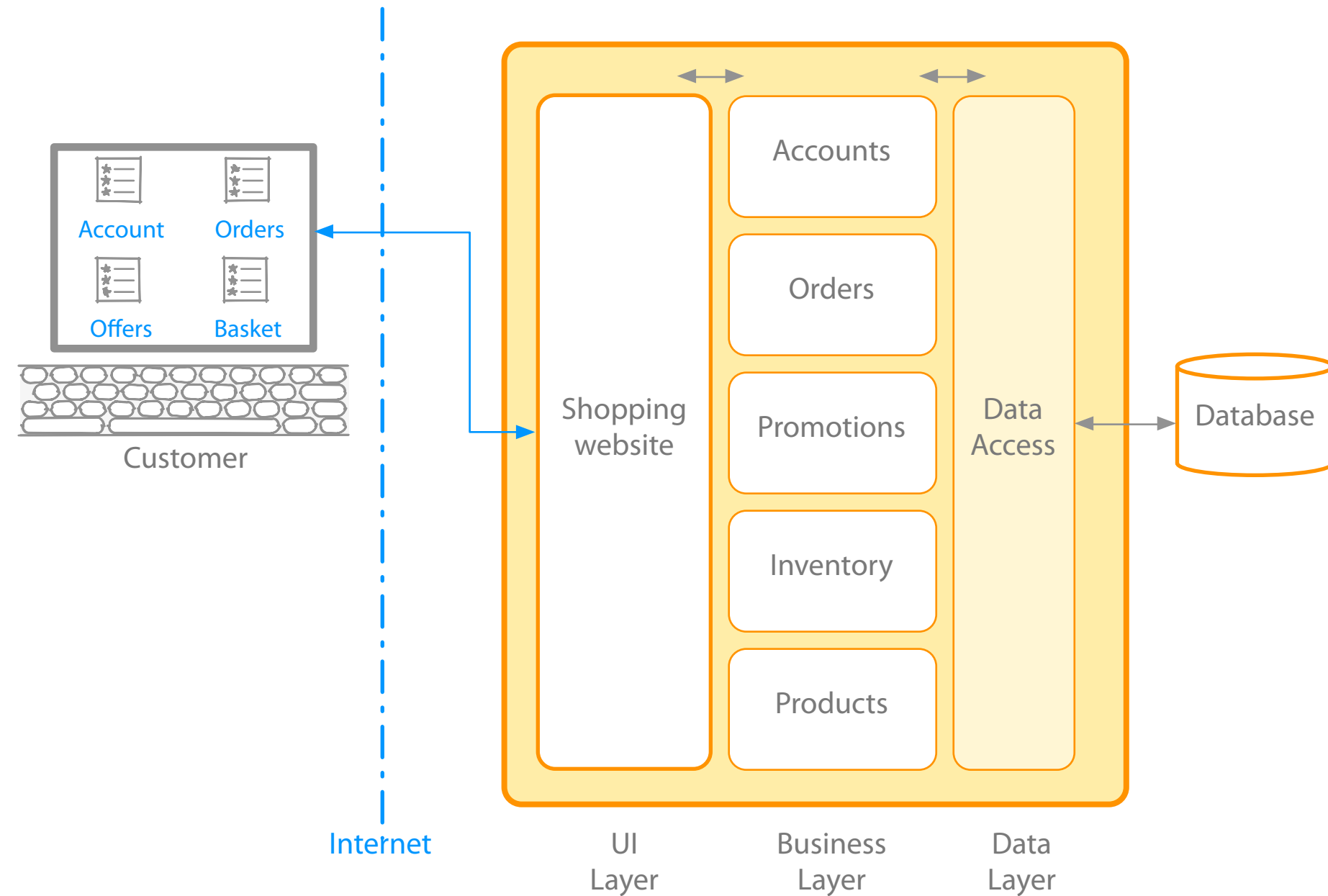
Scaling requires duplication of the whole

Single service on server

Minor change could result in complete rebuild

Easy to replicate environment

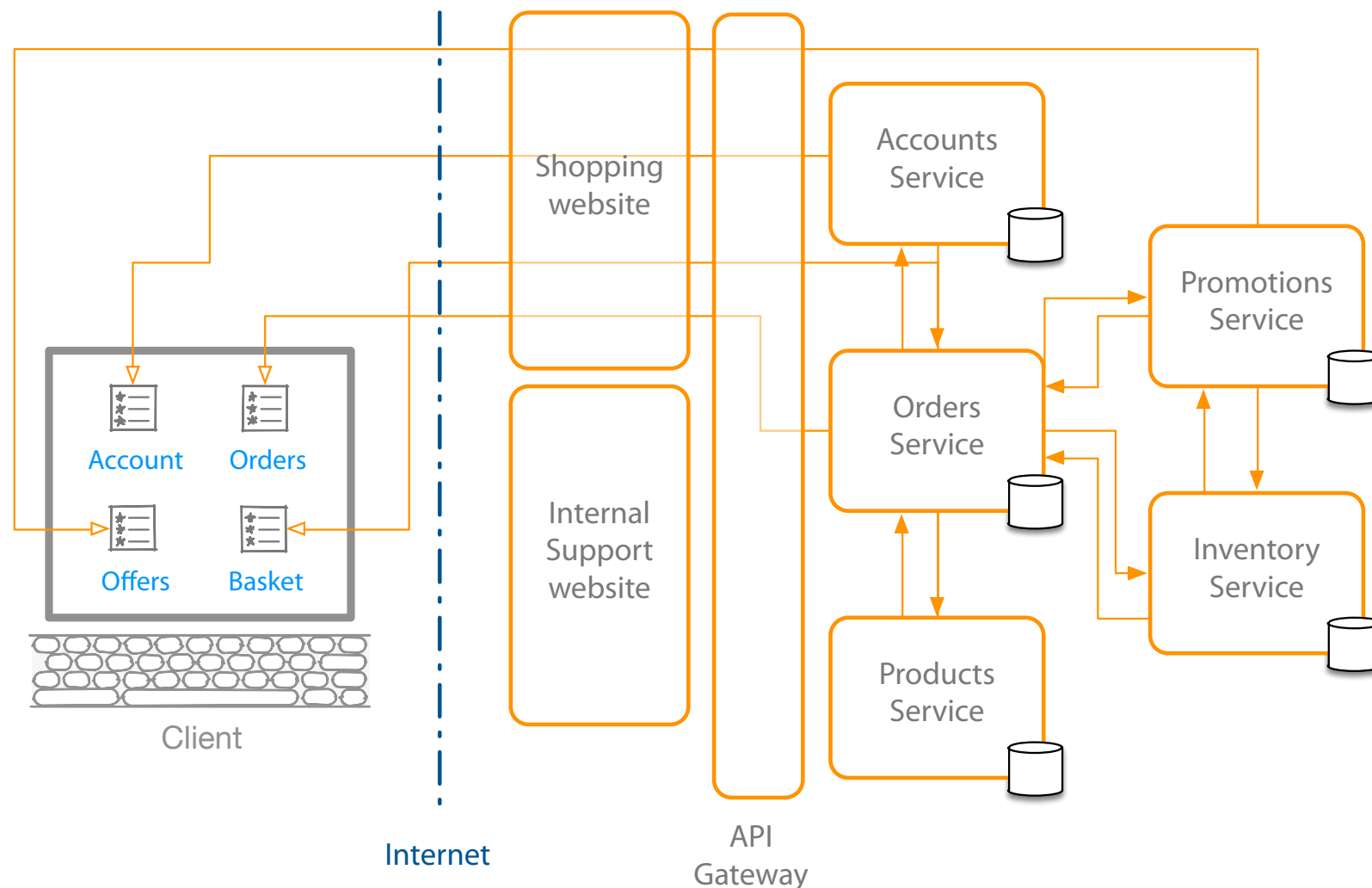
Microservices: The Monolithic



Emergence of Microservices

Why Now? | Benefits

Emergence of Microservices : **Why Now?**



Need to respond to change quickly

Need for reliability

Business domain-driven design

Automated test tools

Release and deployment tools

On-demand hosting technology

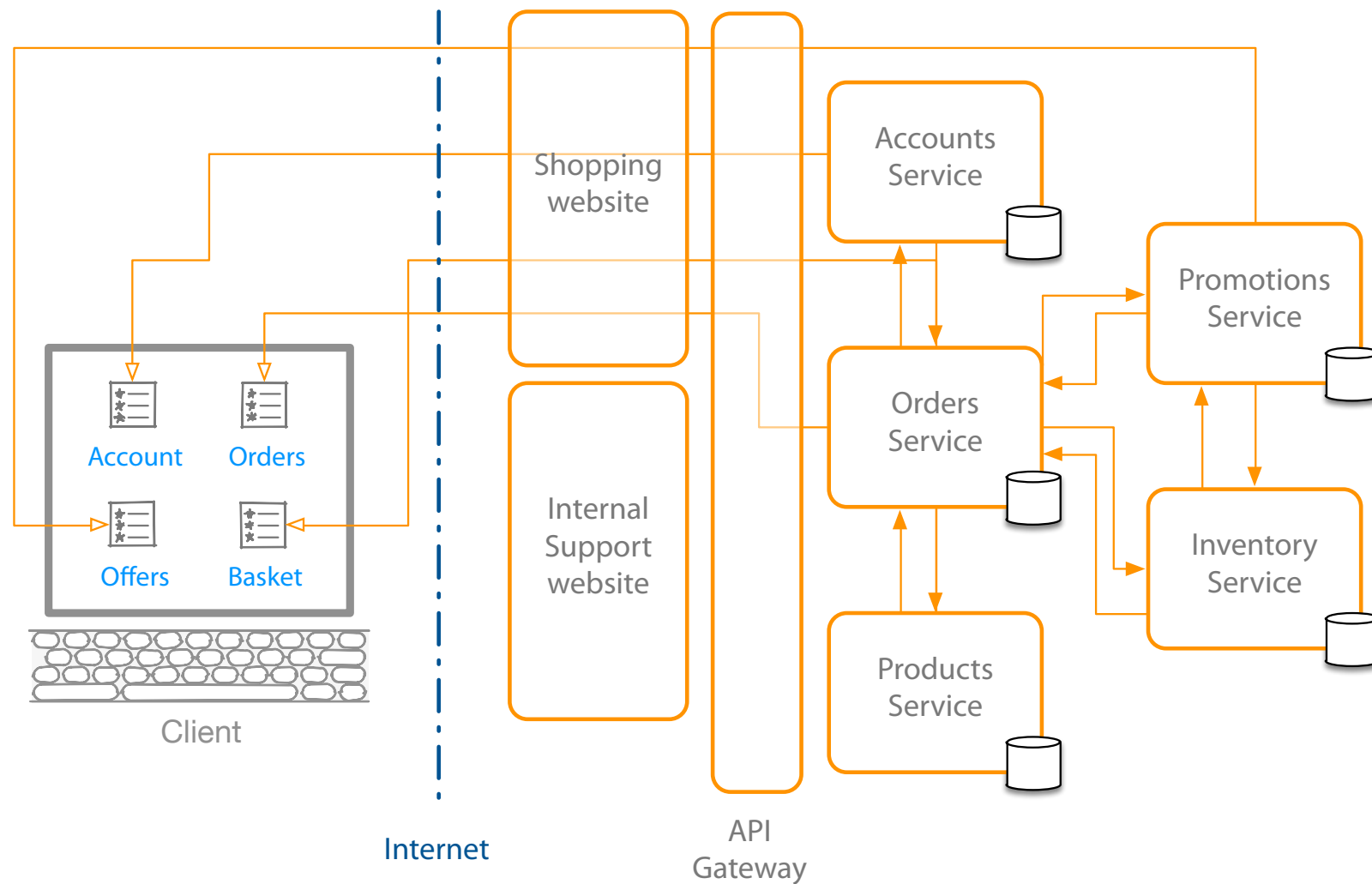
On-line cloud services

Need to embrace new technology

Asynchronous communication technology

Simpler server side and client side technology

Emergence of Microservices : **Benefits**



Shorter development times

Reliable and faster deployment

Enables frequent updates

Decouple the changeable parts

Security

Increased uptime

Fast issue resolution

Highly scalable and better performance

Better ownership and knowledge

Right technology

Enables distributed teams

Microservices Design Principles

Introduction | Principles | Summary

Microservices Design Principles: Introduction

High Cohesion

Autonomous

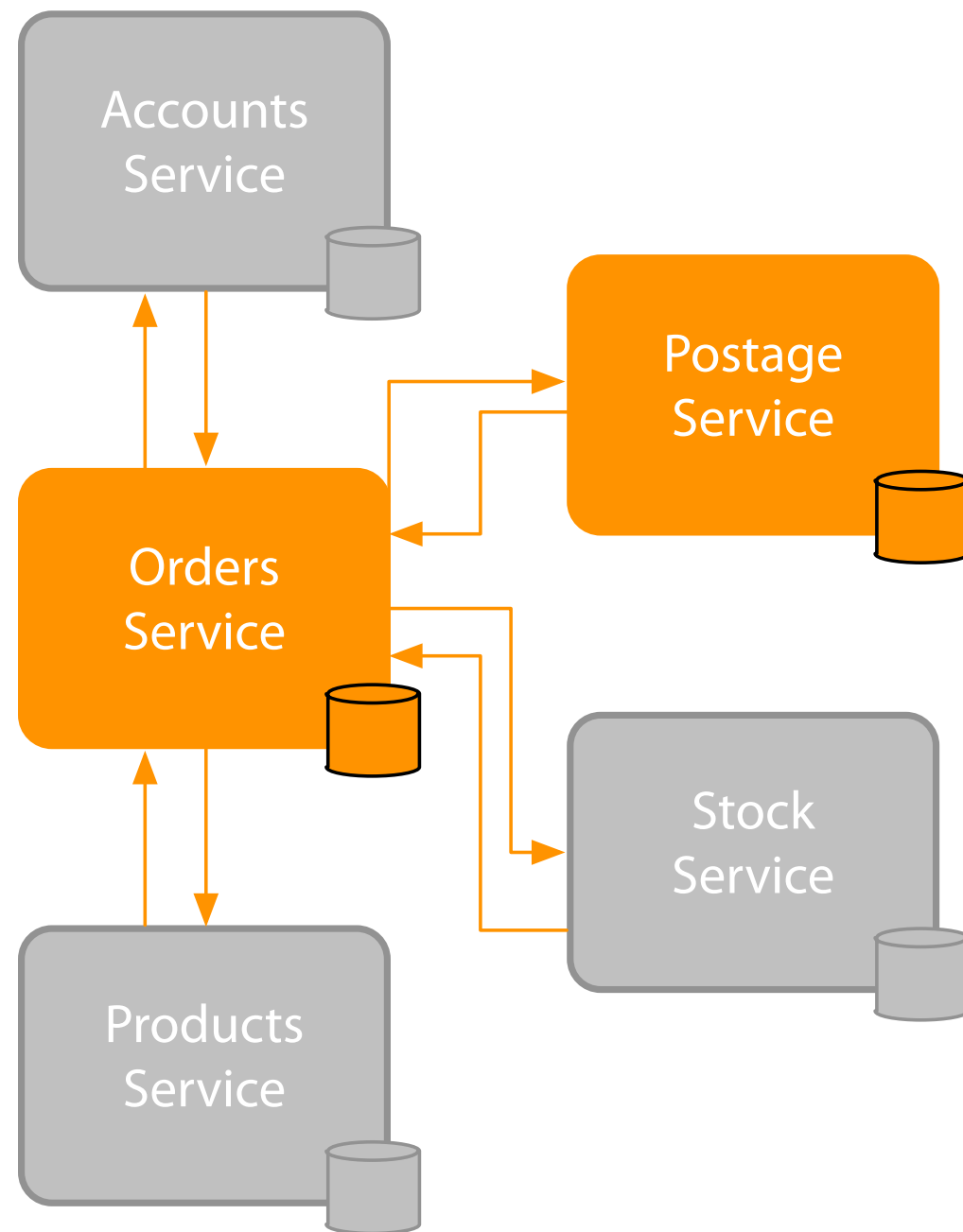
Business Domain
Centric

Resilience

Observable

Automation

Microservices Design Principles: High Cohesion



Single focus

Single responsibility

SOLID principle

Only change for one reason

Reason represents

A business function

A business domain

Encapsulation principle

OOP principle

Easily rewritable code

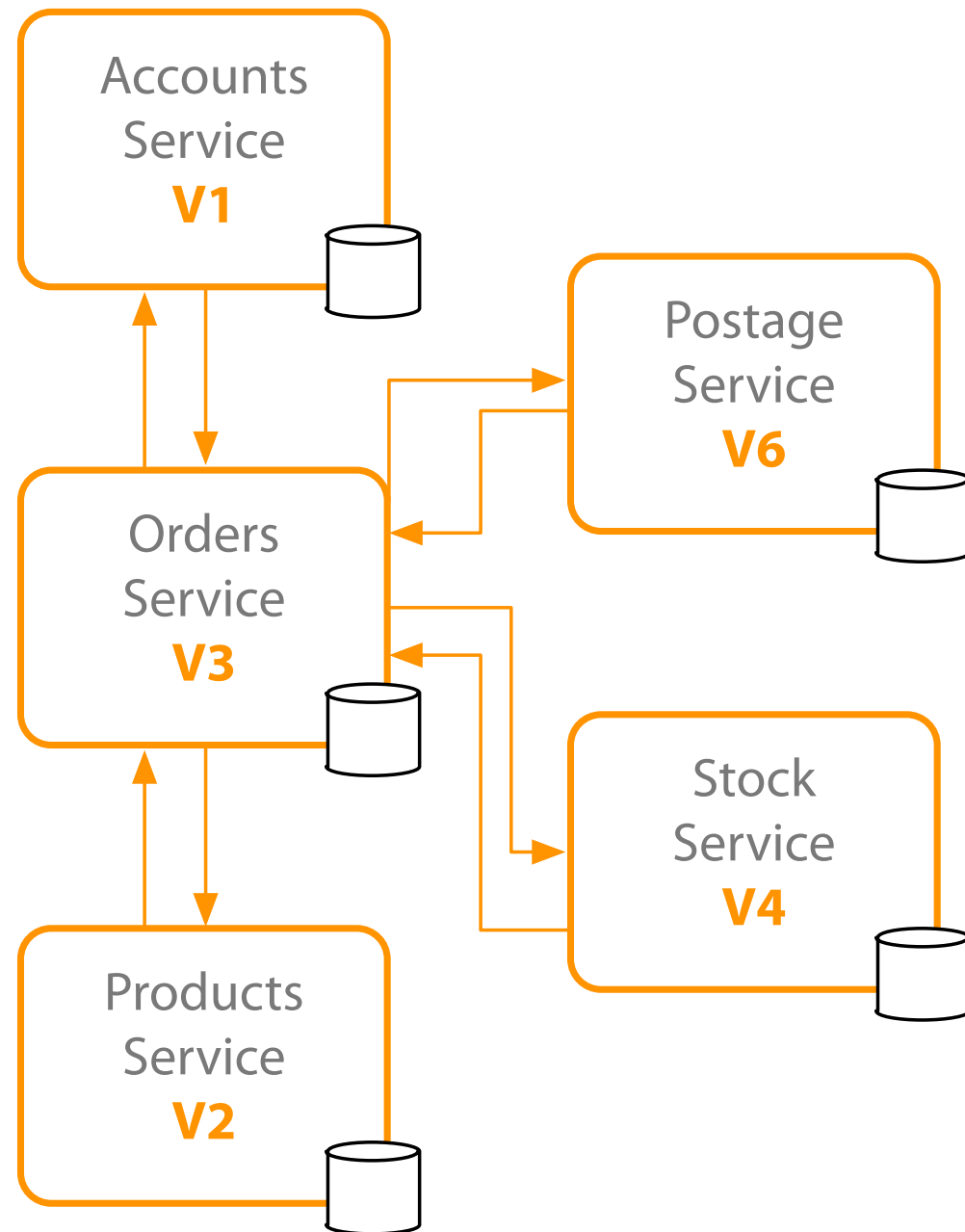
Why

Scalability

Flexibility

Reliability

Microservices Design Principles: **Autonomous**



Loose coupling

Honor contracts and interfaces

Stateless

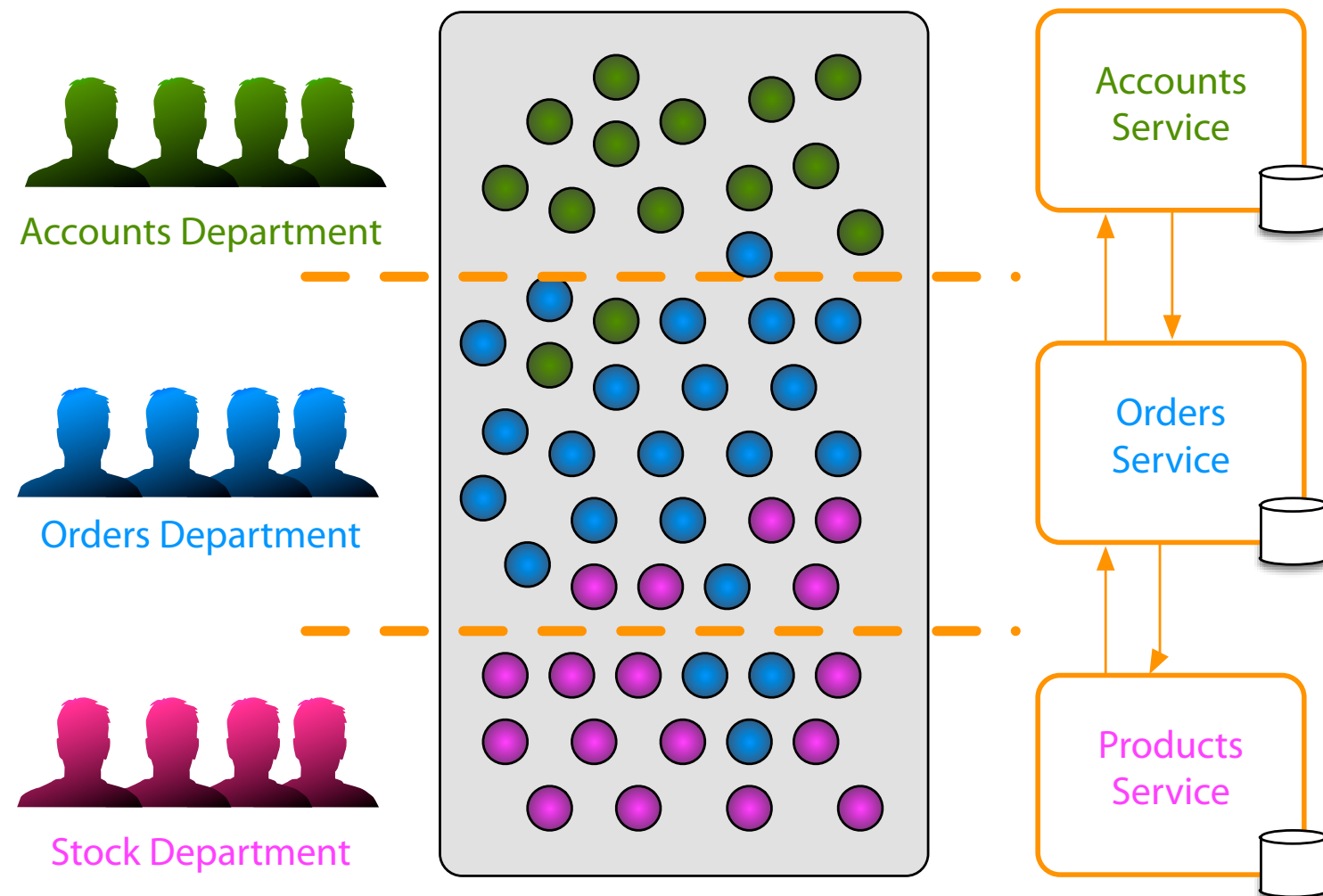
Independently changeable

Independently deployable

Backwards compatible

Concurrent development

Design Principles: Business Domain Centric



Service represents business function

Accounts Department

Postage calculator

Scope of service

Bounded context from DDD

Identify boundaries\seams

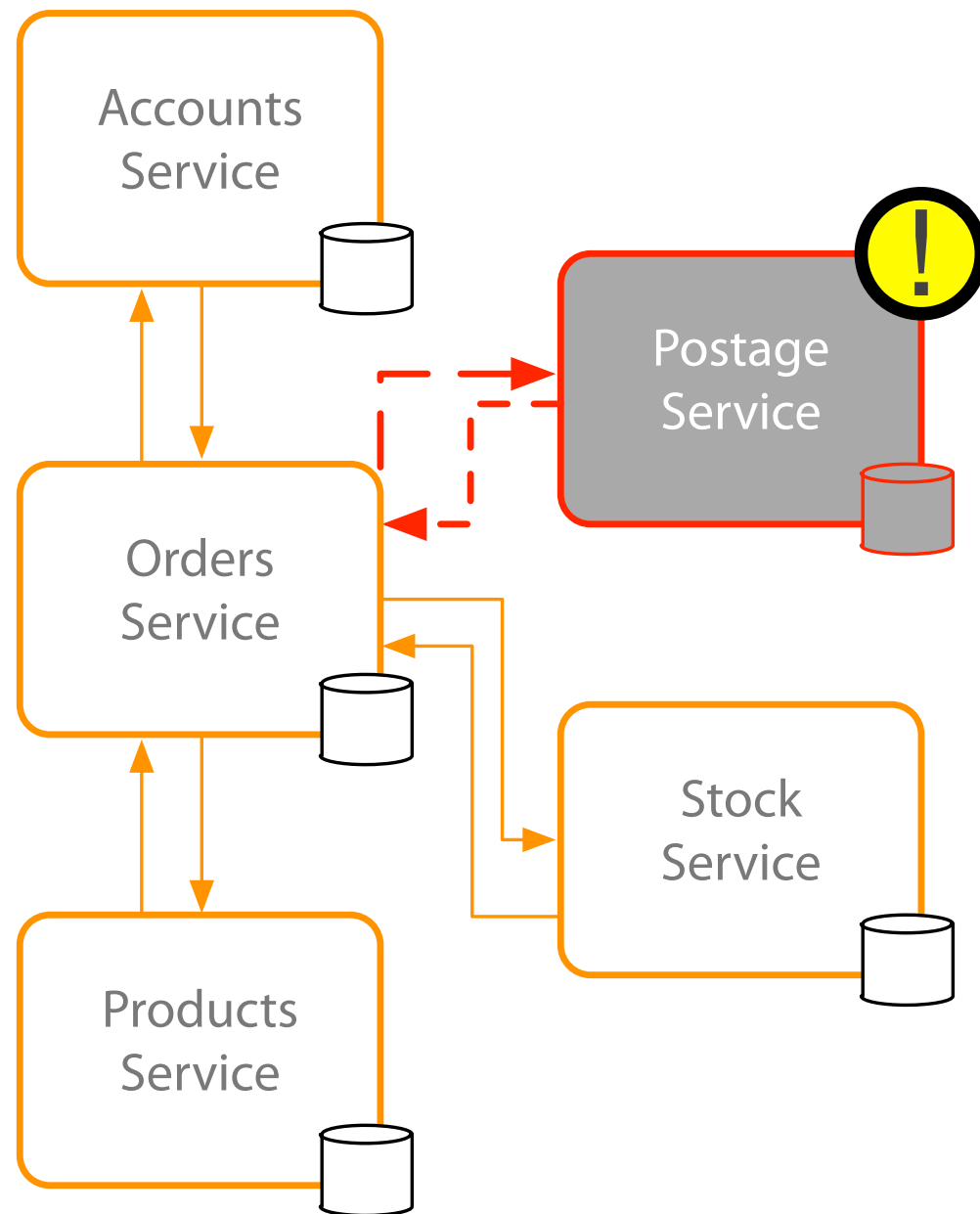
Shuffle code if required

Group related code into a service

Aim for high cohesion

Responsive to business change

Microservices Design Principles: Resilience



Embrace failure

- Another service
- Specific connection
- Third-party system

Degrade functionality

Default functionality

Multiple instances

- Register on startup
- Deregister on failure

Types of failure

- Exceptions\Errors
- Delays
- Unavailability

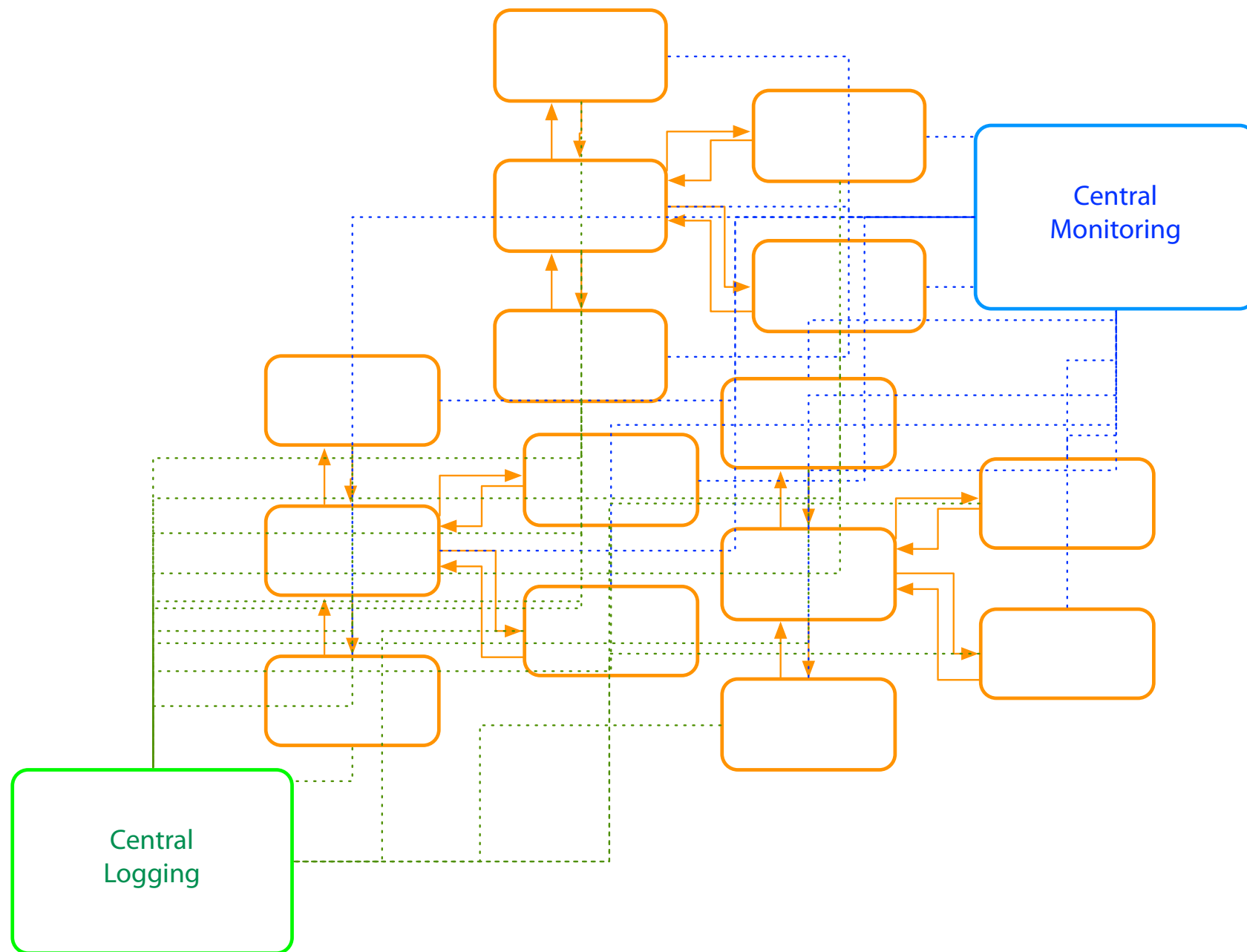
Network issues

- Delay
- Unavailability

Validate input

- Service to service
- Client to service

Microservices Design Principles: **Observable**



System Health

Status

Logs

Errors

Centralized monitoring

Centralized logging

Why

Distributed transactions

Quick problem solving

Quick deployment requires feedback

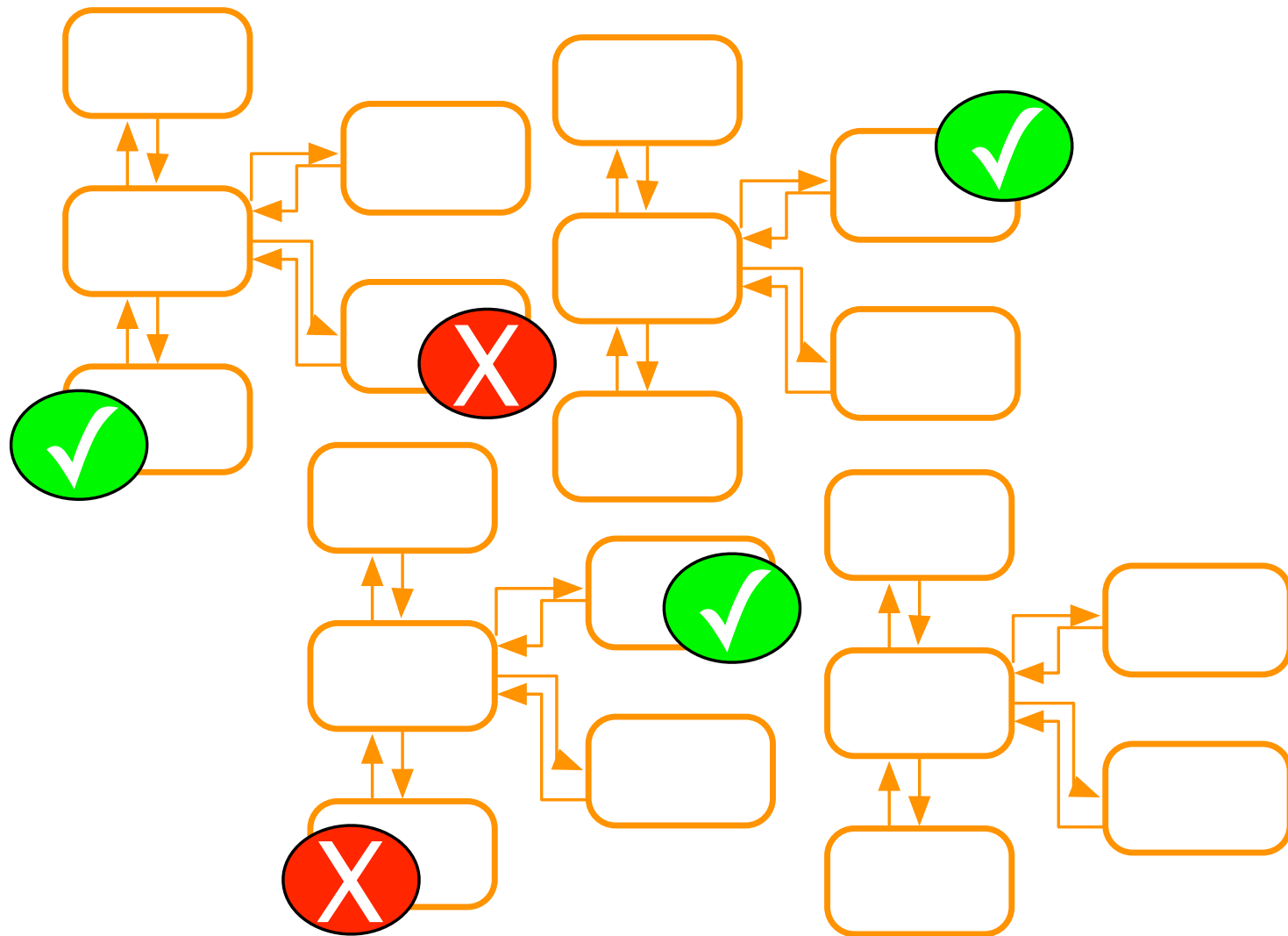
Data used for capacity planning

Data used for scaling

Whats actually used

Monitor business data

Microservices Design Principles: Automation



Tools to reduce testing

- Manual regression testing
- Time taken on testing integration
- Environment setup for testing

Tools to provide quick feedback

- Integration feedback on check in
- Continuous Integration

Tools to provide quick deployment

- Pipeline to deployment
- Deployment ready status
- Automated deployment

- Reliable deployment
- Continuous Deployment

Why

- Distributed system
- Multiple instances of services
- Manual integration testing too time consuming
- Manual deployment time consuming and unreliable

Module Summary



Microservices

Service

Introduction

The Monolithic

Emergence of Microservices

Why Now?

Benefits

Microservices Design Principles

High Cohesion

Autonomous

Business Domain Centric

Resilience

Observable

Automation