IN DEFENSE OF MICROSERVICES

Microservices: From Hype to Doubt

The Era of Hype

Microservices: From Hype to Doubt

The Era of Hype

MICROSERVICES—THE FUTURE OF SOFTWARE ARCHITECTURE

WHY MONOLITHS ARE A THING OF THE PAST

UNLOCK UNLIMITED SCALABILITY WITH MICROSERVICES

Microservices: From Hype to Doubt

The Era of Hype

MICROSERVICES—THE FUTURE OF SOFTWARE ARCHITECTURE

WHY MONOLITHS ARE A THING OF THE PAST

UNLOCK UNLIMITED SCALABILITY WITH MICROSERVICES

Microservices: From Hype to Doubt

The Era of Hype

NETFLIX

amazon

Microservices: From Hype to Doubt

The Era of Hype

NETFLIX

amazon

Microservices: From Hype to Doubt

The Era of Hype

Infinitely scalable

Highly resilient

Built for the cloud era

Microservices: From Hype to Doubt

The Era of Hype

Infinitely scalable

Highly resilient

Built for the cloud era

Microservices: From Hype to Doubt

The Era of Hype

Irony is that I was a big **skeptic** of microservices before I became a big **advocate**!

In Defense of Microservices

Microservices: From Hype to Doubt

The Era of Hype

Rising Skepticism

Microservices: From Hype to Doubt

The Era of Hype

Rising Skepticism

THE MICROSERVICES TRADE-OFFS

MONOLITHS STRIKE BACK

THE HIDDEN COSTS OF MICROSERVICES

Microservices: From Hype to Doubt

The Era of Hype

Rising Skepticism

Over-Engineering	Latency Issues	Data Management Difficulties	Testing Complexities

Microservices: From Hype to Doubt

The Era of Hype

Rising Skepticism

Over-Engineering	Latency Issues	Data Management Difficulties	Testing Complexities

Microservices: From Hype to Doubt

The Era of Hype

Rising Skepticism

Microservices: From Hype to Doubt

The Era of Hype

Rising Skepticism

Was adopting microservices the right choice for our organization?

Microservices: From Hype to Doubt

The Era of Hype

Rising Skepticism

Do we have the necessary infrastructure and expertise to manage this complexity?

Microservices: From Hype to Doubt

The Era of Hype

Rising Skepticism

Microservices: From Hype to Doubt

The Era of Hype

Rising Skepticism

Losing Faith

Microservices: From Hype to Doubt

The Era of Hype

IS IT TIME TO ABANDON MICROSERVICES?

Rising Skepticism

THE MICROSERVICE RETROSPECTIVE: WAS IT WORTH IT?

Losing Faith

EMBRACING THE MODULAR MONOLITH

Microservices: From Hype to Doubt

The Era of Hype

IS IT TIME TO ABANDON MICROSERVICES?

Rising Skepticism

THE MICROSERVICE RETROSPECTIVE: WAS IT WORTH IT?

Losing Faith

EMBRACING THE MODULAR MONOLITH

Microservices: From Hype to Doubt

The Era of Hype

Rising Skepticism

Losing Faith

Microservices: From Hype to Doubt

The Era of Hype

Rising Skepticism

Losing Faith

- Overwhelming Complexity
- Skills Gaps
- Insufficient Tooling
- Cultural Misalignment

The Era of Intelligent Maturity

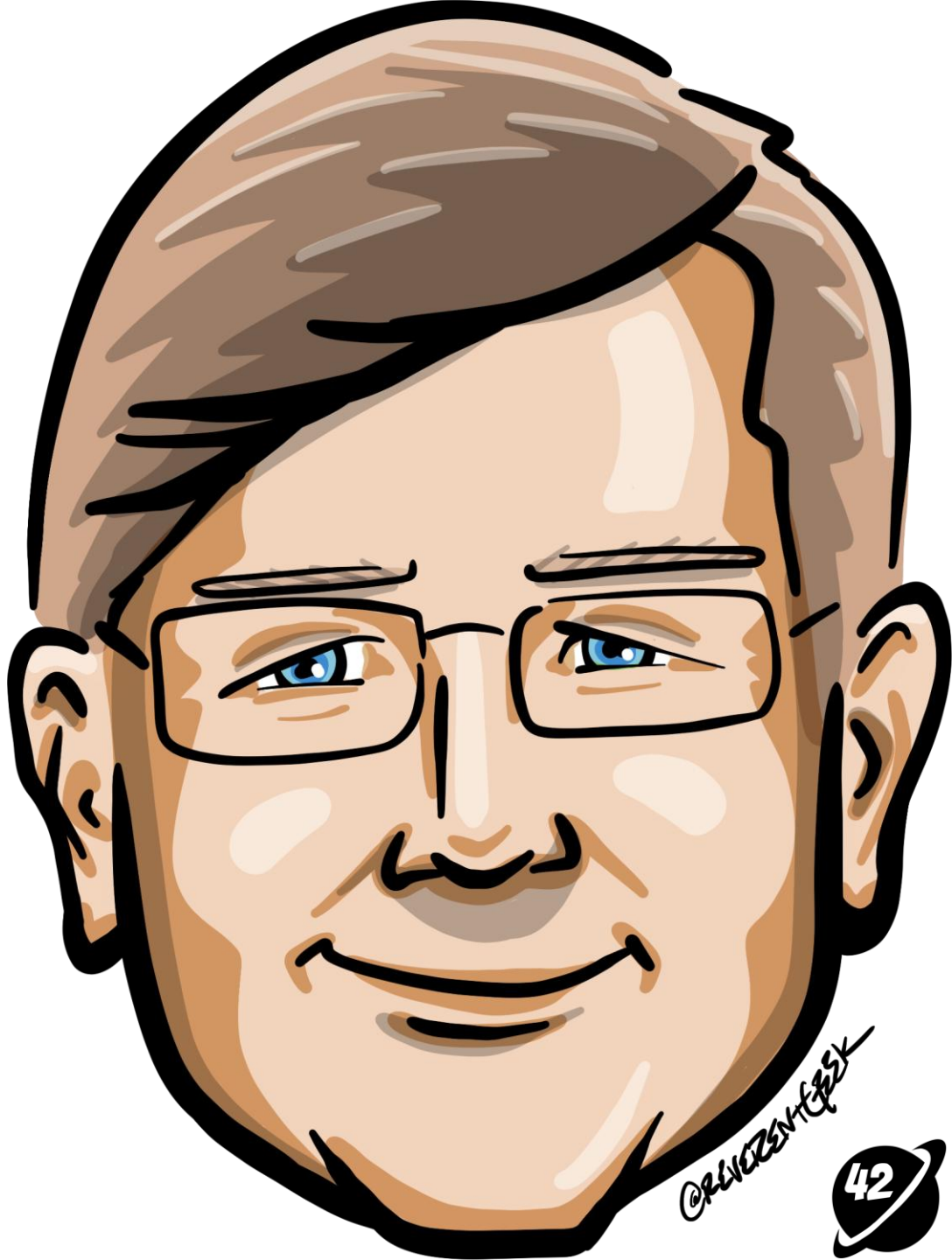
Agenda

- The Microservices Journey
- Comparing Architectural Styles
- Understanding the Core Benefits of Microservices
- Addressing Common Critiques
- Strategies for Successful Microservice Implementation
- Choosing the “Correct” Architectural Style
- Summary and Key Takeaways

Who is Chad Green?

- chadgreen@chadgreen.com
- TaleLearnCode
- ChadGreen.com
- ChadGreen & TaleLearnCode
- ChadwickEGreen

Microsoft® MVP Most Valuable Professional



Comparing Architectural Styles

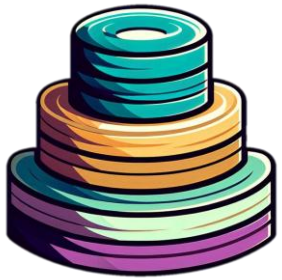
In Defense of Microservices

Introduction to Architectural Styles

A set of principles and patterns guiding
the structure of software systems.

Scalability | **Maintainability** | **Team Productivity**

Monoliths



N-Tier Architectures

Microservices

Modular Monoliths

Monoliths

Single Unified Application: All components are interconnected and interdependent within one codebase.

Advantages

- Simplicity
- Performance
- Ease of Debugging

Challenges

- Scalability Limitations
- Rigid Deployments
- Maintenance Difficulties
- Tight Coupling

Monoliths

Monoliths

N-Tier Architecture

Layered Application Structure: The application is divided into logical layers or tiers (e.g., Presentation, Business Logic, Data Access).

Advantages

- Separation of Concerns
- Reusability
- Manageability

Challenges

- Tight Coupling Between Layers
- Scalability Constraints
- Potential Latency Issues

N-Tier Architecture

Microservices

Collection of Small, Independent Services: Each service focus on a specific business capability; services communicate over well-defined APIs.

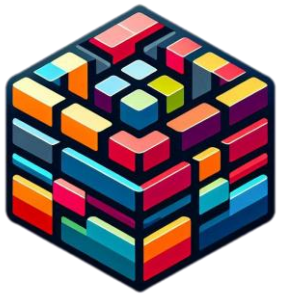
Advantages

- Scalability
- Flexibility in Technology Stack
- Independent Deployments
- Fault Isolation

Challenges

- Operational Complexity
- Distributed System Challenges
- Data Consistency
- Testing Difficulties

Microservices



Modular Monoliths

Monolith with Modular Design: Single application divided into distinct, independent modules which communicate internally within the application

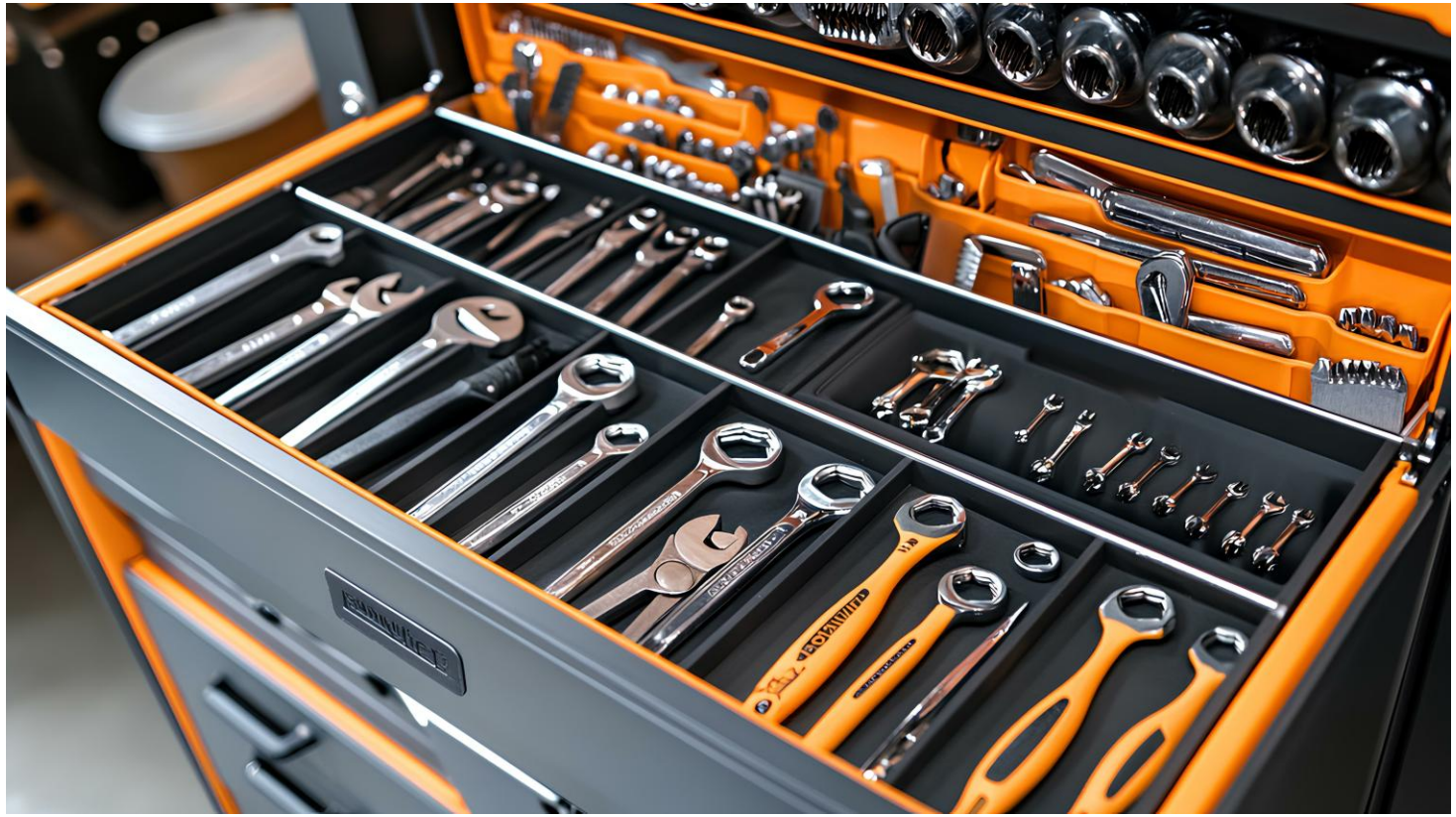
Advantages

- Maintainability
- Performance
- Simpler Deployment
- Easier Transition

Challenges

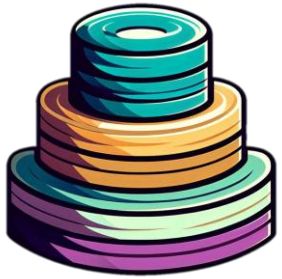
- Scalability Limitations
- Deployment Constraints
- Potential Module Erosion
- Limited Fault Isolation

Modular Monoliths



Introduction to Architectural Styles

Monoliths



N-Tier Architectures

Microservices

Modular Monoliths

The Core Benefits of Microservices

In Defense of Microservices

Scalability

- Independent Scaling
- Elasticity

Fault Tolerance

Service Isolation

Resilience

Independent Deployments

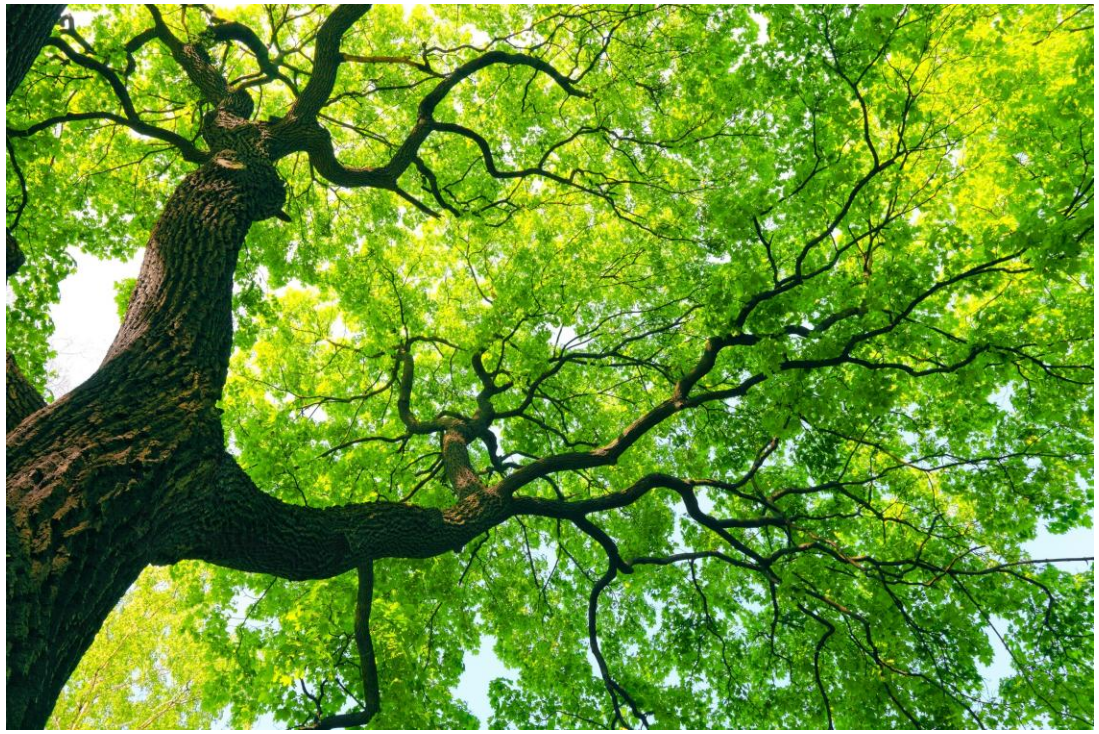
- Faster Release Cycles
- Reduced Deployment Risk
- Flexibility in Scheduling

Team Autonomy

Autonomous Teams

Parallel Development

Faster Decision Making



Flexibility in Technology Choices

- Polyglot Programming
- Innovation
- Reduce Technology Lock-In

Alignment with Business Goals

Organizing Around Business Capabilities

Autonomous Teams

Business Agility

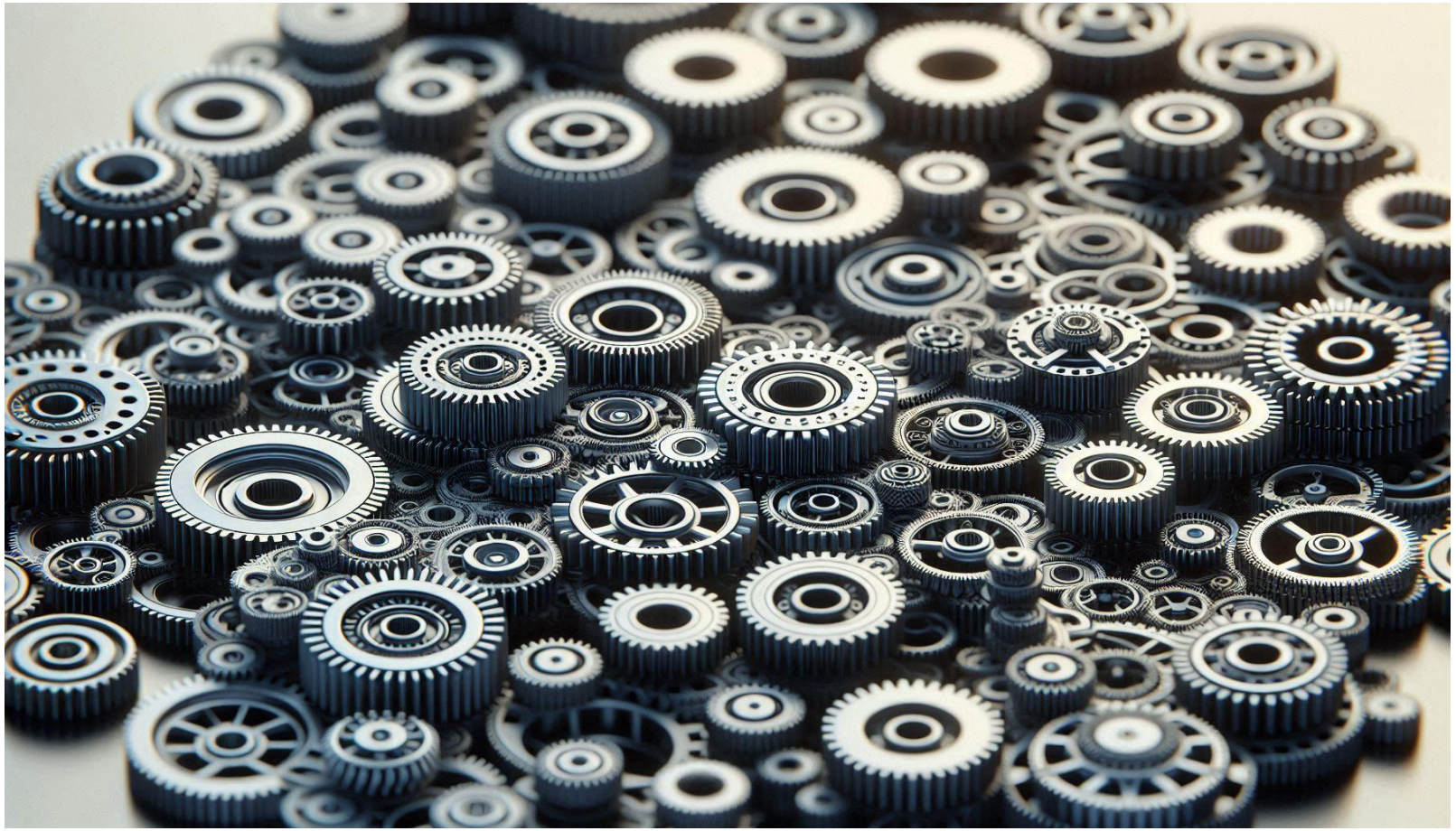
Core Benefits of Microservices

- Scalability
- Fault Tolerance
- Independent Deployments
- Team Autonomy
- Flexibility in Technology Choices
- Alignment with Business Goals

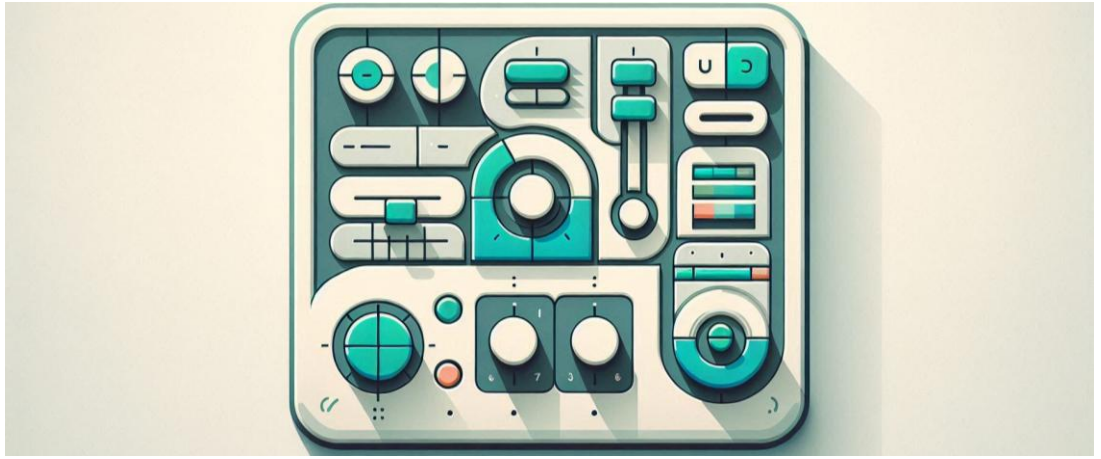
Addressing Common Critiques

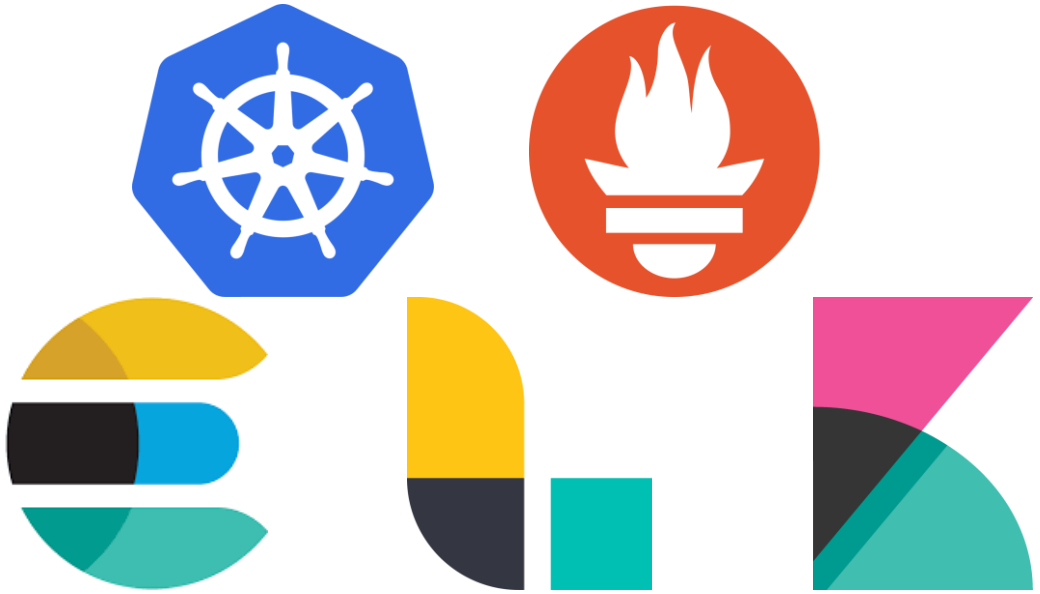
In Defense of Microservices

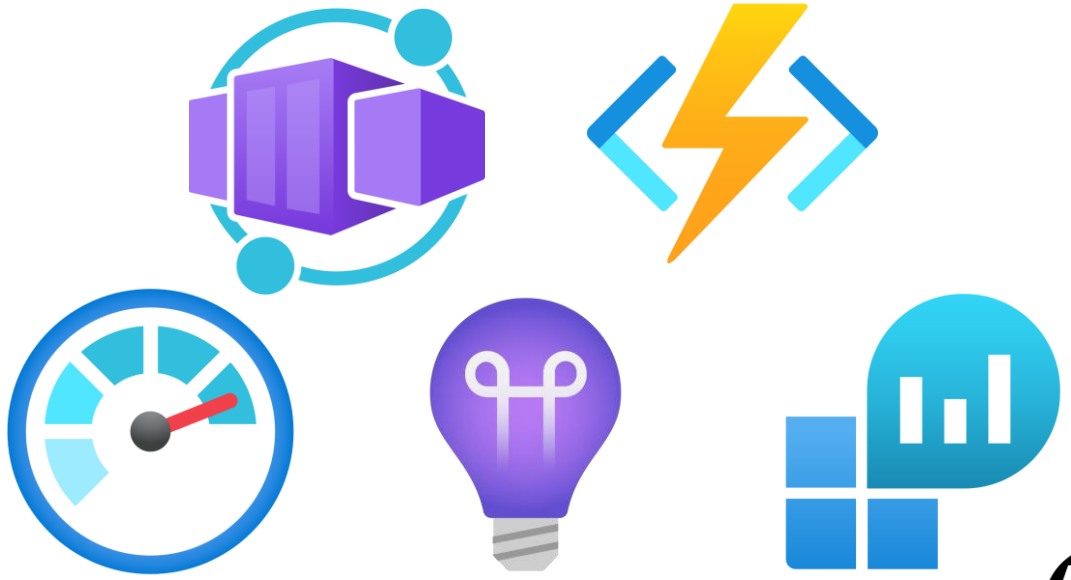
Operational Complexity



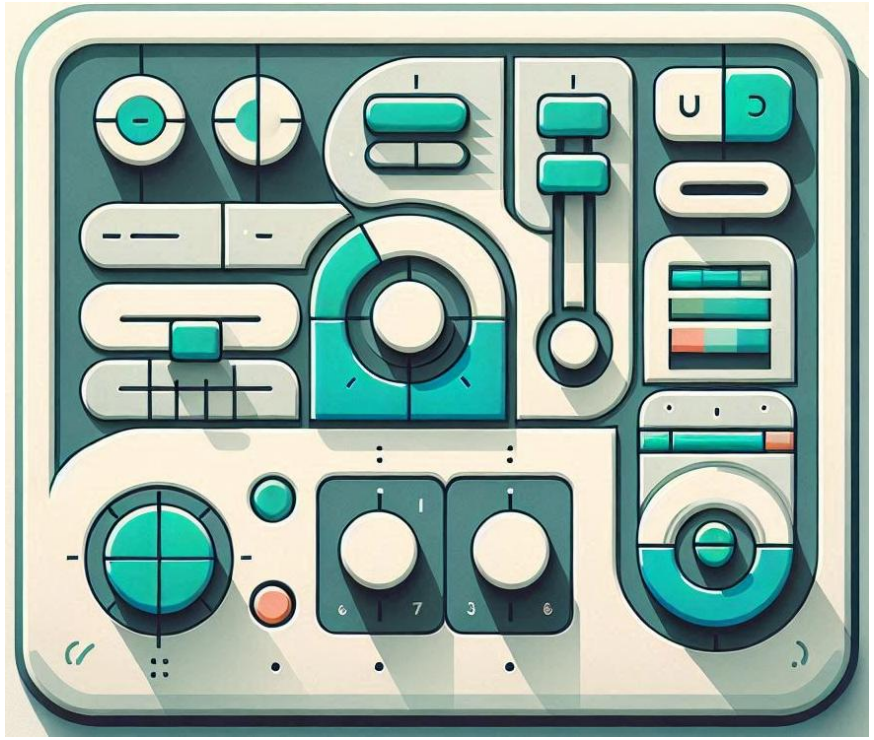
Operational Complexity

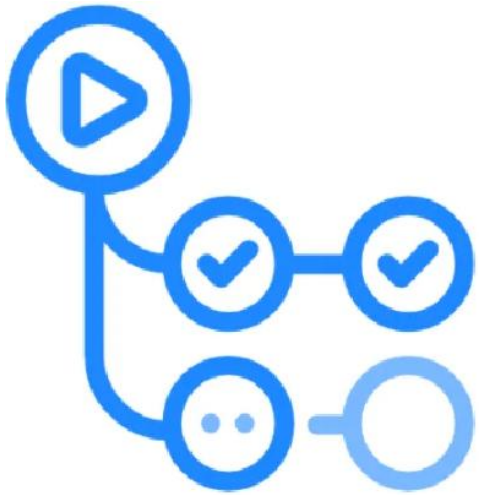






Operational Complexity



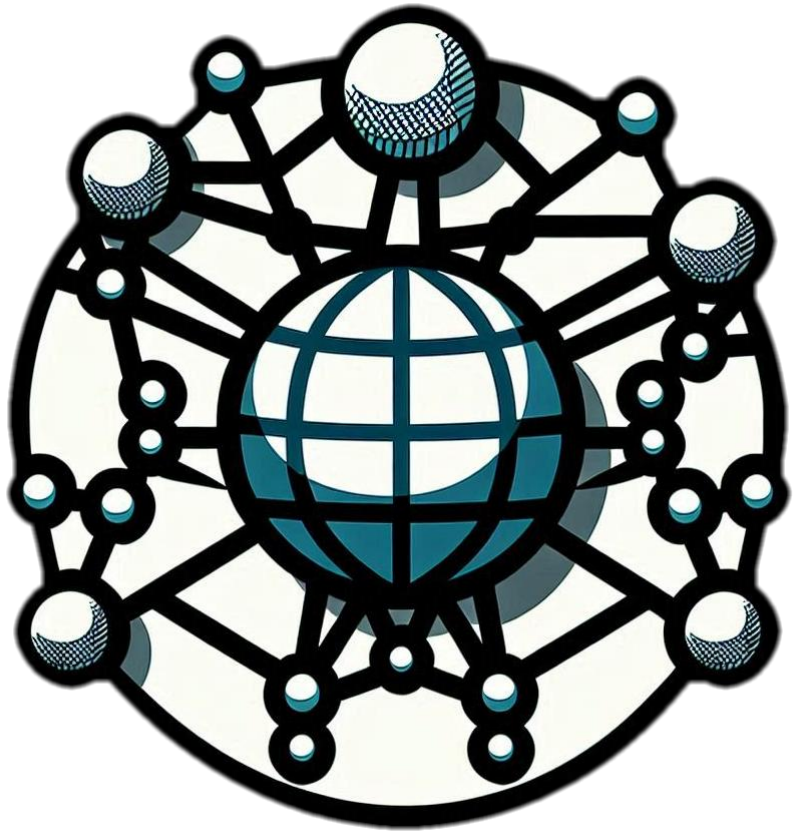


Distributed Systems Challenges

Distributed Systems Challenges

The Challenges

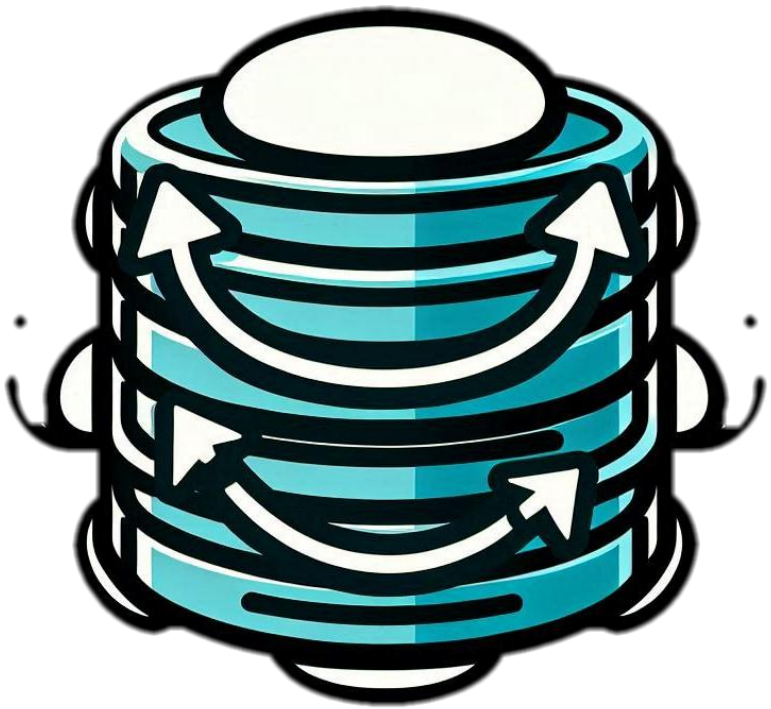
- Network Complexity



Distributed Systems Challenges

The Challenges

- Network Complexity
- Data Consistency



Distributed Systems Challenges

The Challenges

- Network Complexity
- Data Consistency

Mitigation Strategies

- Asynchronous Communication

Distributed Systems Challenges

The Challenges

- Network Complexity
- Data Consistency

Mitigation Strategies

- Asynchronous Communication
- Retry Policies and Circuit Breakers

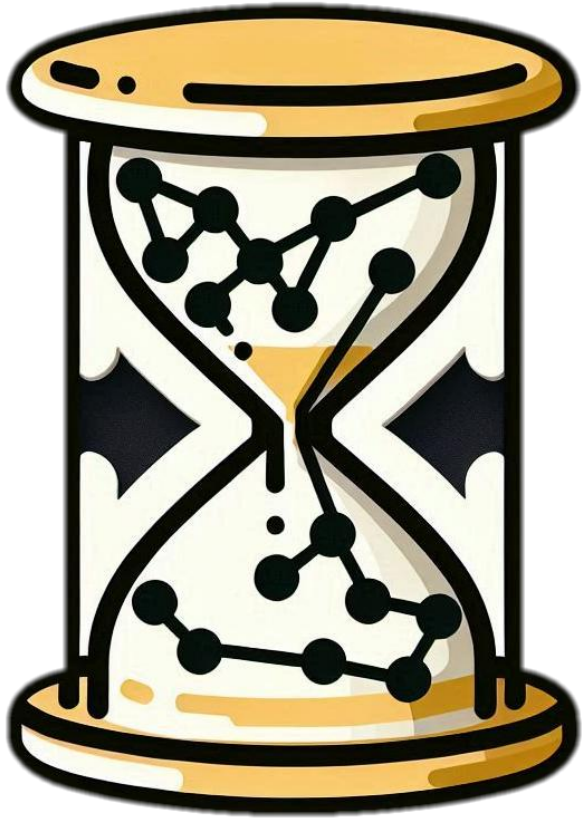
Distributed Systems Challenges

The Challenges

- Network Complexity
- Data Consistency

Mitigation Strategies

- Asynchronous Communication
- Retry Policies and Circuit Breakers
- Eventual Consistency



Distributed Systems Challenges

The Challenge

- Network Complexity
- Data Consistency

Mitigation Strategies

- Asynchronous Communication
- Retry Policies and Circuit Breakers
- Eventual Consistency

Data Management and Consistency

Data Management and Consistency

The Challenges

- Data Consistency Across Services
- Distributed Transactions

Data Management and Consistency

The Challenges

- Data Consistency Across Services
- Distributed Transactions

Strategies for Managing Data Consistency

- Embracing Eventual Consistency

Data Management and Consistency

The Challenges

- Data Consistency Across Services
- Distributed Transactions

Strategies for Managing Data Consistency

- Embracing Eventual Consistency
- Saga Pattern

Data Management and Consistency

The Challenges

- Data Consistency Across Services
- Distributed Transactions

Strategies for Managing Data Consistency

- Embracing Eventual Consistency
- Saga Pattern
- Event Sourcing

Data Management and Consistency

The Challenges

- Data Consistency Across Services
- Distributed Transactions

Strategies for Managing Data Consistency

- Embracing Eventual Consistency
- Saga Pattern
- Event Sourcing
- Leveraging Tools and Frameworks

Testing Complexities

Testing Complexities

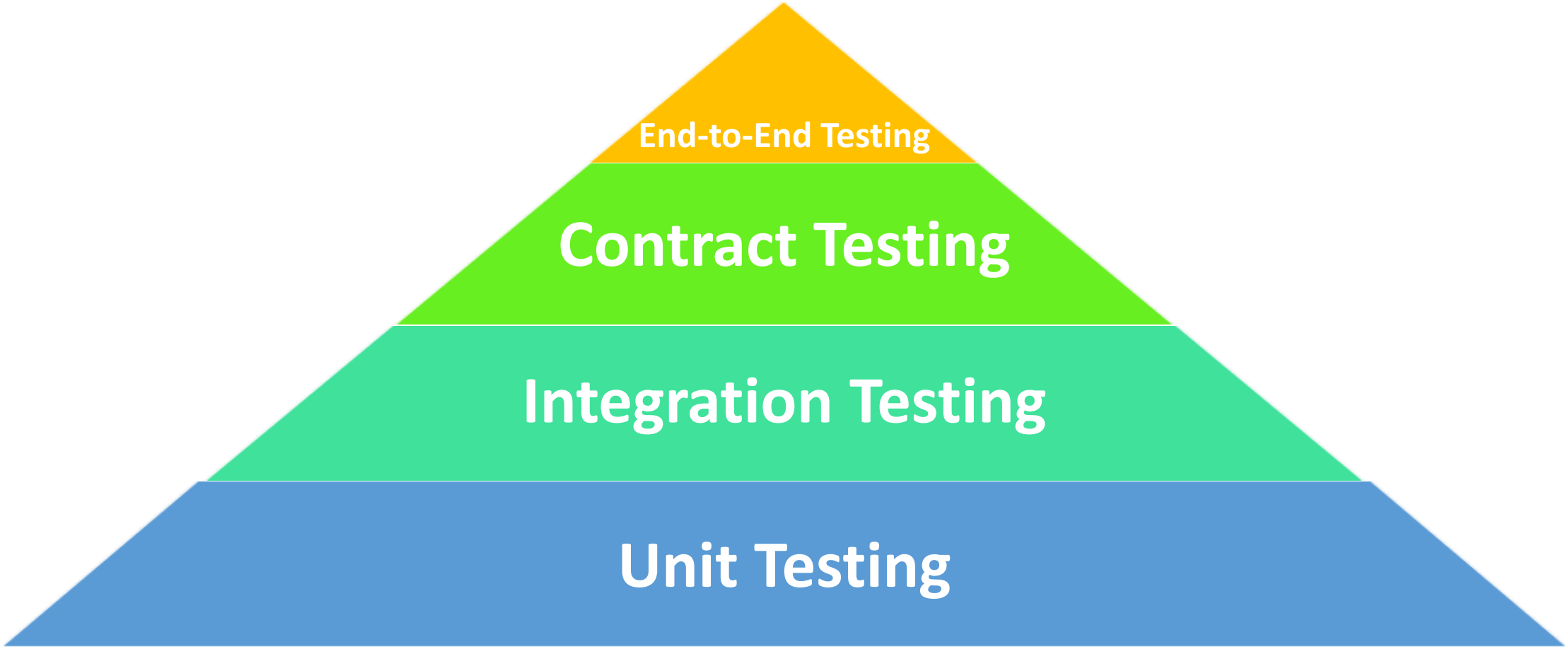
The Challenges

- Increased Complexity in End-to-End Testing
- Need for Extensive Integration and Contract Testing

Opportunities and Advantages

- Improved Unit Testing
- Enhanced Confidence Through Testing

Testing Pyramid



Testing Complexities

The Challenges

- Increased Complexity in End-to-End Testing
- Need for Extensive Integration and Contract Testing

Opportunities and Advantages

- Improved Unit Testing
- Enhanced Confidence Through Testing

Skill and Expertise Requirements

Skill and Expertise Requirements

The Challenges

- Higher Level of Expertise Required
- Learning Curve

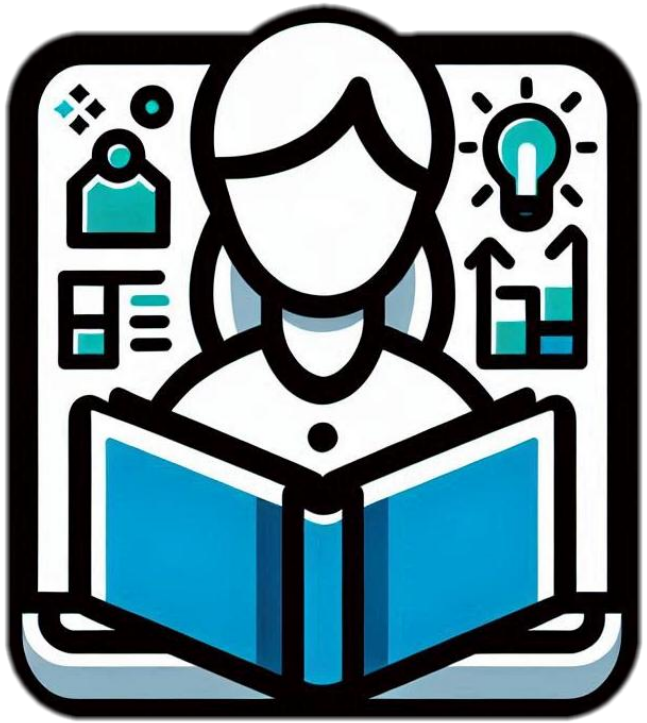
Skill and Expertise Requirements

The Challenges

- Higher Level of Expertise Required
- Learning Curve

Mitigation Strategies

- Embracing Continuous Learning



Skill and Expertise Requirements

The Challenges

- Higher Level of Expertise Required
- Learning Curve

Mitigation Strategies

- Embracing Continuous Learning
- Leveraging External Resources

Addressing Common Critiques

- Operational Complexity
- Distributed System Challenges
- Data Management and Consistency
- Testing Complexity
- Skills and Expertise Requirements

Strategies for Successful Microservice Implementation

In Defense of Microservices

Culture and Organizational Strategies

Culture and Organizational Strategies

Embracing DevOps Culture

Culture and Organizational Strategies

- Foster Collaboration
- Shared Responsibility
- Automate Processes

Fostering Continuous Learning

Culture and Organizational Strategies

- Promote Professional Development
- Building a Learning Culture
- Stay Current with Industry Trends

Aligning Teams with Business Capabilities

Culture and Organizational Strategies

- Organize Around Business Domains
- Enhance Communication
- Empower Autonomous Teams

Culture and Organizational Strategies

Embracing DevOps Culture

Fostering Continuous Learning

Aligning Teams with Business Capabilities

Technology Best Practices

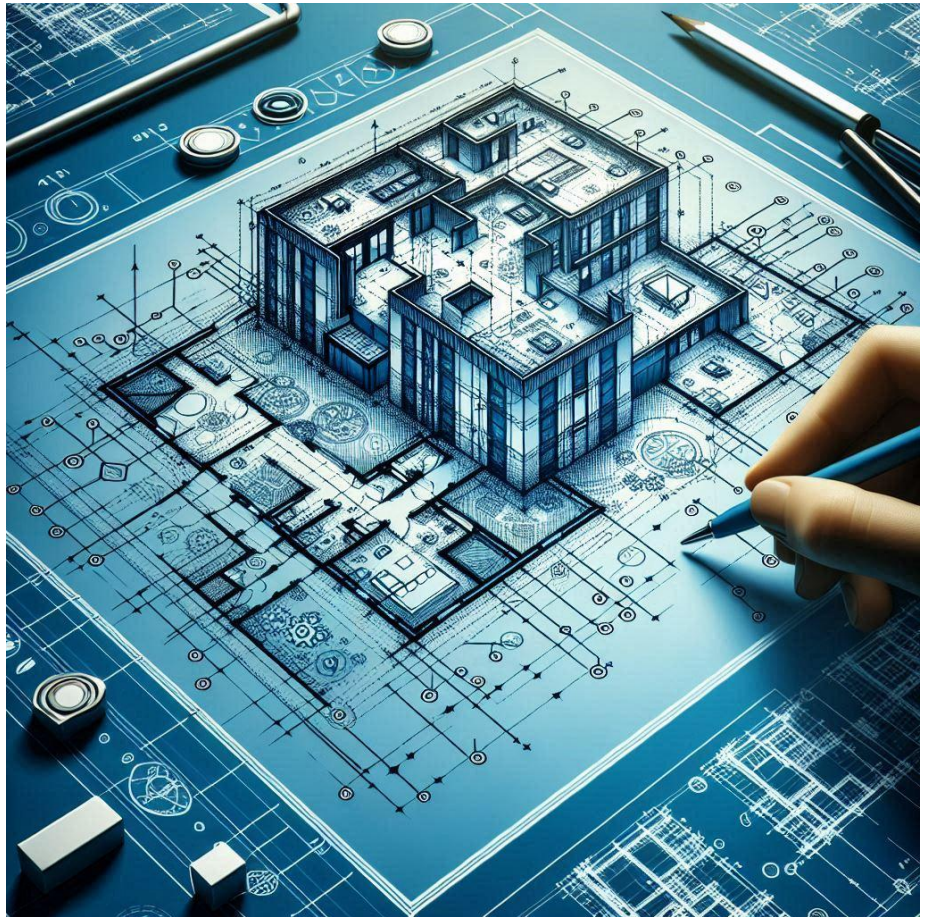
Technology Best Practices

- Clear Architectural Vision
- Effective Testing Strategies
- Monitoring and Overserving

Clear Architectural Vision

Technology Best Practices

- Define Service Boundaries
- Develop a Comprehensive Plan
- Communicate the Vision



Effective Testing Strategies

Technology Best Practices

- Comprehensive Testing Approach
- Automate Testing in CI/CD Pipelines
- Focus on Critical Paths

Monitoring and Observability

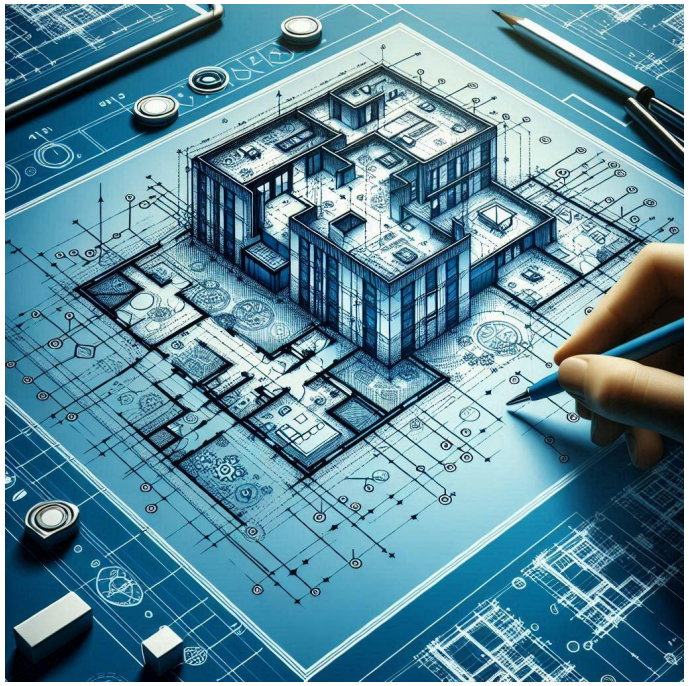
Technology Best Practices

- Centralize Logging and Monitoring
- Set up Alerting and Dashboards
- Enable Proactive Issue Resolution

Technology Best Practices

- Clear Architectural Vision
- Effective Testing Strategies
- Monitoring and Overserving

Technology Best Practices

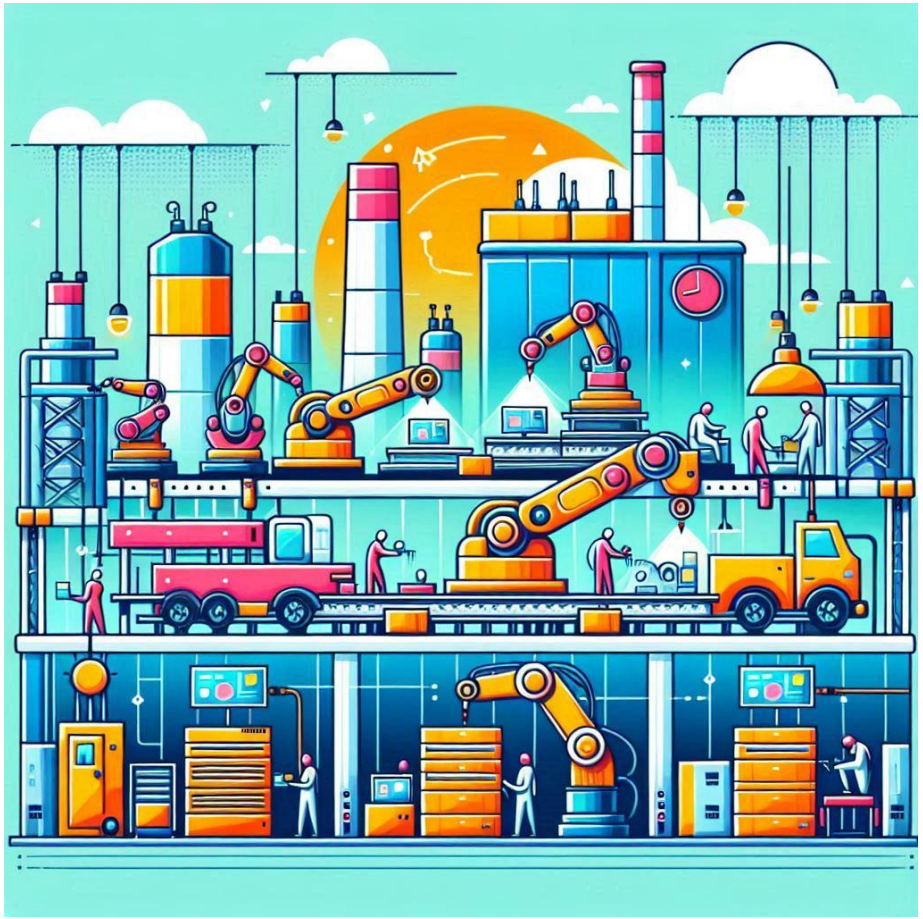


Clear Architectural Vision

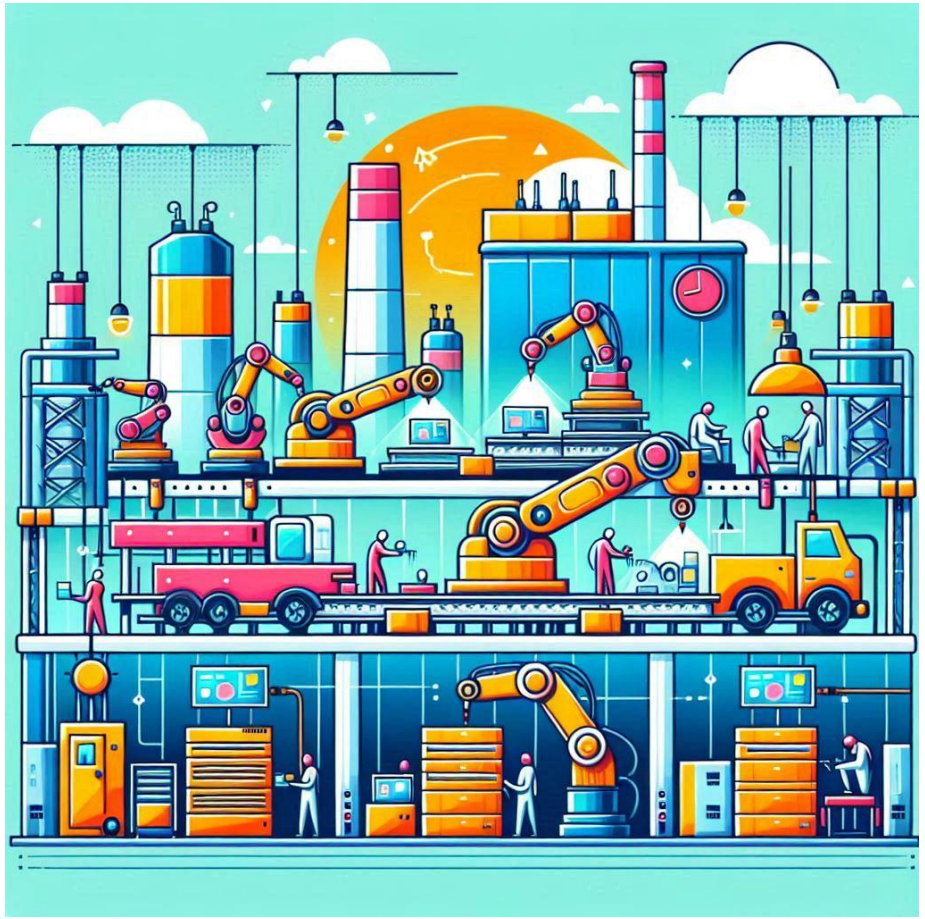
Effective Testing Strategies

Monitoring and Observability

Process and Tooling



Process and Tooling

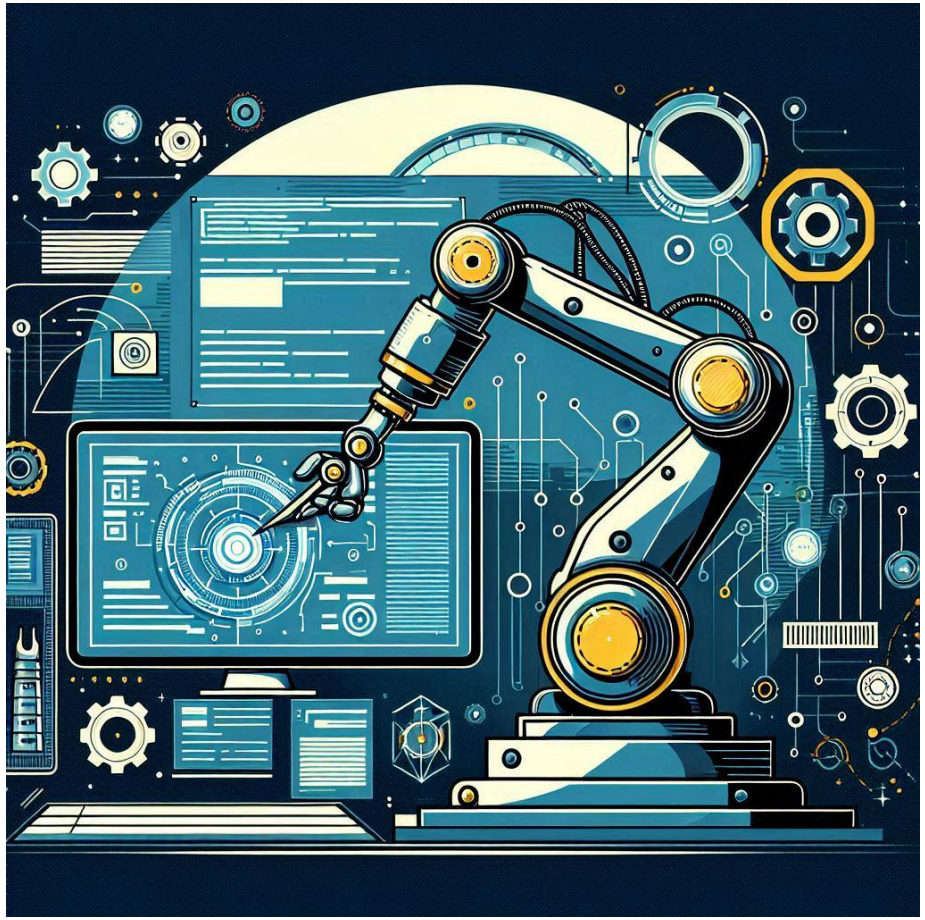


- Clear Architectural Vision
- Effective Testing Strategies
- Monitoring and Overserving

Automation and IaC

Technology Best Practices

- Automate Infrastructure Provisioning
- Streamline Deployments
- Consistency and Repeatability



Containerization and Orchestration

Technology Best Practices

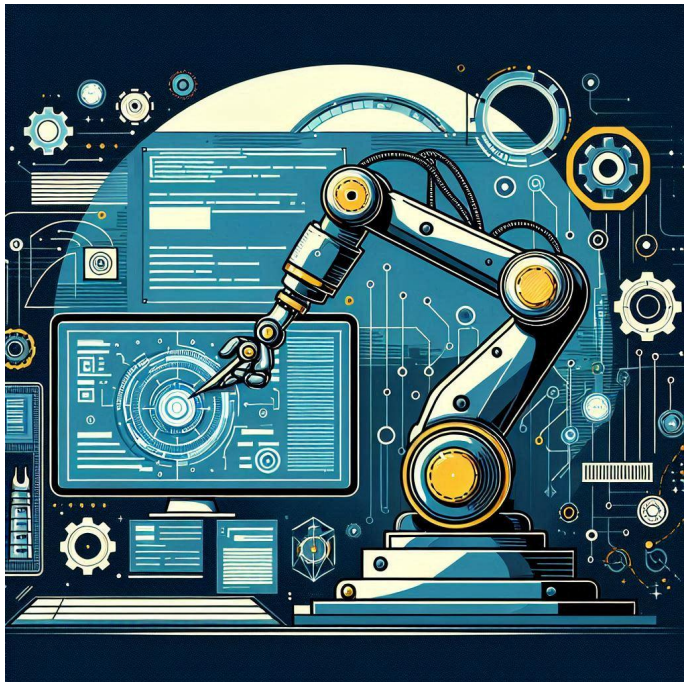
- Containerize Service
- Orchestrate with Kubernetes
- Serverless Options

Appropriate Tools & Technologies

Technology Best Practices

- Select the Right Stack
- Leverage Cloud Services
- Implement Observability Tools

Process and Tooling



Automation and IaC

Containerization and Orchestration

Appropriate Tools and Technologies

Choosing the “Correct” Architectural Style

In Defense of Microservices

Comparison of Architectural Styles

Criterion	Monolith	N-Tier	Modular Monolith	Microservices
Scalability	Limited	Moderate	Improved	High
Development Speed	Faster for small projects	Moderate	Faster for medium complexity projects	Faster for large, complex projects
Deployment	Slower	Moderate	Moderate	Faster
Maintenance	Complex	Moderate	Easier	Easier
Complexity	Simpler for small projects	Moderate	Moderate	High
Fault Isolation	Limited	Moderate	Improved	High

Comparison of Architectural Styles

Criterion	Monolith	N-Tier	Modular Monolith	Microservices
Scalability	Limited	Moderate	Improved	High
Development Speed	Faster for small projects	Moderate	Faster for medium complexity projects	Faster for large, complex projects
Deployment	Slower	Moderate	Moderate	Faster
Maintenance	Complex	Moderate	Easier	Easier
Complexity	Simpler for small projects	Moderate	Moderate	High
Fault Isolation	Limited	Moderate	Improved	High

Comparison of Architectural Styles

Criterion	Monolith	N-Tier	Modular Monolith	Microservices
Scalability	Limited	Moderate	Improved	High
Development Speed	Faster for small projects	Moderate	Faster for medium complexity projects	Faster for large, complex projects
Deployment	Slower	Moderate	Moderate	Faster
Maintenance	Complex	Moderate	Easier	Easier
Complexity	Simpler for small projects	Moderate	Moderate	High
Fault Isolation	Limited	Moderate	Improved	High

Comparison of Architectural Styles

Criterion	Monolith	N-Tier	Modular Monolith	Microservices
Scalability	Limited	Moderate	Improved	High
Development Speed	Faster for small projects	Moderate	Faster for medium complexity projects	Faster for large, complex projects
Deployment	Slower	Moderate	Moderate	Faster
Maintenance	Complex	Moderate	Easier	Easier
Complexity	Simpler for small projects	Moderate	Moderate	High
Fault Isolation	Limited	Moderate	Improved	High

Comparison of Architectural Styles

Criterion	Monolith	N-Tier	Modular Monolith	Microservices
Scalability	Limited	Moderate	Improved	High
Development Speed	Faster for small projects	Moderate	Faster for medium complexity projects	Faster for large, complex projects
Deployment	Slower	Moderate	Moderate	Faster
Maintenance	Complex	Moderate	Easier	Easier
Complexity	Simpler for small projects	Moderate	Moderate	High
Fault Isolation	Limited	Moderate	Improved	High

Real-World Use Cases

Monolith

Early-Stage Startups

Startup developing a basic e-commerce platform

Real-World Use Cases

Monolith	N-Tier
Early-Stage Startups	Enterprise Applications

Banking application needing scalability and maintainability

Real-World Use Cases

Monolith	N-Tier	Modular Monolith
Early-Stage Startups	Enterprise Applications	Medium-Sized Businesses

Online retail company with separate modules for inventory management, order processing, and customer service

Real-World Use Cases

Monolith	N-Tier	Modular Monolith	Microservices
Early-Stage Startups	Enterprise Applications	Medium-Sized Businesses	Large Tech Companies

Handle immense load and complexity of delivering services to millions of users worldwide

Summary and Key Takeaways

In Defense of Microservices

Key Takeaways

Microservices Overview

Addressing Common Critiques

Strategies for Successful Implementation

Choosing the “Correct” Architectural Style

Call for Action

Adopt Microservices Thoughtfully

Invest in Continuous Learning

Leveraging Tools and Best Practices

Join the Conversation

Thank You

chadgreen@chadgreen.com

TaleLearnCode

ChadGreen.com

ChadGreen & TaleLearnCode

ChadwickEGreen