Introduction to Microservices

In today's hyper-competitive business environment, a microservices architecture can help your organization stand out—enabling you to scale applications while improving cycle times. This infographic provides an at-a-glance overview of this exciting approach to software development.

What are microservices?

- An architectural and organizational approach to software development
- Software is composed of small independent services that communicate over well-defined APIs—with each service owned by small, self-contained teams
- Make applications easier to scale and faster to develop
- Enable innovation and accelerate time-to-market for new features

Monolithic vs. microservices

- All processes are tightly coupled and run as a single service—making it complex to add or improve an application's features as the code base grows
- Complexity limits experimentation and innovation
- Added risk of single process failure

Monolithic architectures

- Applications are built as independent components that run each application process as a service
- Each service performs a single function
- Services are less complex to update, deploy, and scale

Microservices architectures

- Each component service can be developed, deployed, operated, and scaled without affecting the functioning of other services
- No need to share code or implementation with other services
- Any communication between individual components happens via well-defined APIs

They are autonomous

- Each service is designed for a set of capabilities focused on solving a specific problem
- Services can be broken into smaller services if complexity occurs over time

They are specialized

Benefits of microservices

Why adopt a microservices approach? Your organization can experience the following benefits:

Agility

- Small, independent teams are empowered to work more independently and quickly
- Shorter development cycle times
- Better aggregate throughput

Easy Deployment

- Continuous integration and continuous delivery
- Enables experimentation
- Easier to update code
- Accelerates time-to-market for new features

Reusable Code

- Use functions for multiple purposes—e.g., as a building block for other features
- Enables creation of new capabilities without writing code from scratch

Flexible Scaling

- Each service is independently scaled to meet demand for the application feature it supports
- Teams can right-size infrastructure needs, accurately measure feature costs, and maintain availability

Resilience

- Increased resistance to failure
- Applications handle total service failure by degrading functionality and not crashing the entire application

Technological Freedom

- Teams can choose the best tool to solve their specific problems
- No need for one-size-fits-all solutions

Characteristics of microservices

Monolith vs. microservices architectures

- All processes are tightly coupled and run as a single service—making it complex to add or improve an application's features as the code base grows
- Complexity limits experimentation and innovation
- Added risk of single process failure

Monolithic architectures

- Applications are built as independent components that run each application process as a service
- Each service performs a single function
- Services are less complex to update, deploy, and scale

Microservices architectures

- Each component service can be developed, deployed, operated, and scaled without affecting the functioning of other services
- No need to share code or implementation with other services
- Any communication between individual components happens via well-defined APIs

They are autonomous

- Each service is designed for a set of capabilities focused on solving a specific problem
- Services can be broken into smaller services if complexity occurs over time

They are specialized

Book your FREE Consultation Today

HATCHWORKS.COM

+1 404 267 1991

BOOK NOW