Amach: Transforming Monolithic Applications to Microservices

Enabling airlines, travel, and e-commerce brands to achieve agility, scalability, and resilience by decomposing legacy monoliths into independent, API driven services.

Problem Statement

Legacy monolithic systems bundle user interface, business logic, and data access into one large codebase. This tight coupling leads to slow-release cycles, resource inefficiencies, and singlepoint failures. Modern business demands including rapid feature delivery, unpredictable traffic spikes, and high availability expose these weaknesses, making it difficult to innovate or scale effectively.

Key Issues:	Problem	Solution
Deployment Rigidity	Large codebase requires coordinated, waterfall style releases, delaying time to market.	Adopt agile CI/CD pipelines to deploy individual services independently.
Scalability Constraints	Entire application must scale even if only one component is under heavy demand.	Enable per service scaling so compute resources match actual traffic patterns.
Resilience Risks	Failure in any module brings the entire system down	Isolate failures by service, with circuit breakers and retries to contain and recover.
Data Management Complexity	Single database schema couples all functionality, complicating changes and scaling.	Decentralize data stores per domain, using event driven patterns for eventual consistency.
Operational Overhead	Monitoring, logging, and securing a monolith is simpler in fewer pieces—but less flexible.	Implement centralized observability (logs, metrics, tracing) and automated orchestration (e.g., Kubernetes).

Solution Overview

Phased Microservices Adoption

Assess & Plan
• Analyze existing monolith modules, dependencies, and bottlenecks.
• Define bounded contexts and prioritize domains for initial extraction.

Design Communication & Data
• Define REST/gRPC APIs or asynchronous messaging channels.
• Decouple databases perservice.

Implement Observability & Resilience
• Deploy centralized logging (ELK/Fluentd), metrics (Prometheus), and distributed tracing (Jaeger).
• Integrate circuit breakers, bulkheads, & retries to isolate failures & maintain uptime.

Extract Independent Services
• Identify lowcoupling modules (e.g., authentication, billing).
• Extract first microservice using the Strangler Pattern, routing traffic via an API.

Build DevOps & Automation
• Containerize services with Docker; orchestrate with Kubernetes or ECS.
• Establish CI/CD pipelines with automated testing, security scans, and deployments.

Scale & Govern
• Apply perservice autoscaling policies based on realtime metrics.
• Enforce service level agreements via API gateways, authentication/authorization (OAuth, JWT), and secrets management.

Key Features of Microservices Modernization

- **Independent Deployments**
- **PerService Scaling**
- **Fault Isolation**
- **Polyglot Flexibility**
- **Centralized Observability**
- **Automated Resilience**

Benefits & Business Impact

Faster Time to Market
Shorter release cycles deploy features in days instead of months.

Greater Agility
Teams iterate rapidly on individual services to respond to business needs.

Optimized Costs
Rightsize resources perservice; pay only for what you use.

Scalability on Demand
Scale only the services under load, ensuring performance during spikes.

Improved Resilience
Failures are isolated, reducing time to recovery and improving availability.

Technology Freedom
Adopt best-of-breed languages and data stores per service.

Ready To Learn More?

Modernize your legacy systems with a microservices architecture. Achieve operational agility, cost efficiency, and robust resilience backed by **Amach's** deep cloud expertise and best practice frameworks.

Contact Us

14 Clanwilliam Square, Dublin
amach.com
sales@amach.com