Essential Rules for a True Microservice Architecture

1. Single Responsibility Per Service

Each microservice must do one thing and do it well. It should own one business capability (e.g., 'User Management', 'Payment').

2. Independent Deployment

Each service must be deployable independently without breaking other services.

3. Separate Data Storage

Each microservice must own its database. No shared database between services.

4. Decentralized Communication

Services must communicate over the network (e.g., HTTP/REST, gRPC, message brokers).

5. Autonomy

A microservice should be self-contained and able to operate, deploy, and scale independently.

6. Statelessness

Services should be stateless when possible. Store state externally.

7. API Contracts

Services must expose clear, stable APIs. Avoid breaking changes.

8. Asynchronous Communication (Where Needed)

Use message brokers or events to decouple services.

9. Observability

Each service must have logging, metrics, tracing, and health checks.

Essential Rules for a True Microservice Architecture

10. Resilience and Fault Tolerance

Use circuit breakers, retries, and timeouts.

11. Scalability

Each service should scale independently.

12. DevOps Ready

Microservices must support CI/CD, Docker, Kubernetes.

13. Service Discovery

Services should locate each other dynamically (e.g., Consul, Kubernetes DNS).

14. Security per Service

Each service must handle authentication, authorization, and secure transport.

15. Versioning

APIs must support backward-compatible versioning.

16. Bounded Contexts (DDD)

Services should be modeled around domain-driven design.

17. Loose Coupling, High Cohesion

Services should minimize external dependencies and be internally focused.

18. Minimal Shared Code

Avoid tight coupling through shared libraries. Prefer duplication over dependency.