Service-Oriented Architecture (SOA)

It is a way of designing software where **functionality is split into independent, reusable services** that communicate over a network. Each service does *one business capability well* and can be developed, deployed, and scaled independently.

Think of SOA like a **company with departments**:

- Each department (service) has a clear job
- They talk via formal contracts (APIs)
- They don't care how other departments are implemented internally

Typical SOA Architecture

```yaml
Client / Consumer
        |
        v
Service Interface (SOAP / REST)
        |
        v
Business Logic Service
        |
        v
Database / External Systems
```

Often includes:

- **Enterprise Service Bus (ESB)** for routing, transformation, security
- **Service Registry** for discovery
- **Centralized governance**

SOA vs Monolithic Architecture

Aspect	Monolith	SOA
Deployment	Single unit	Multiple services
Scalability	Scale whole app	Scale individual services
Flexibility	Low	High
Technology	Single stack	Multiple stacks
Failure impact	High	Isolated

SOA vs Microservices (Important ⚠️)

SOA is **not the same** as microservices.

Feature	SOA	Microservices
Service size	Large	Small
Communication	Often SOAP / ESB	REST / gRPC / events
Data sharing	Can share DB	Each service owns DB
Governance	Centralized	Decentralized
Complexity	Medium	High (but flexible)

👉 **Microservices evolved from SOA** by removing heavy ESBs and central control.

When to Use SOA

- ✅ Large enterprises
- ✅ Legacy system integration
- ✅ Multiple teams across tech stacks
- ✅ Need strong governance and compliance
- ❌ Small startups
- ❌ Simple CRUD apps