UNIT V-MICROSERVICES AND DEVOPS:

Defining Micro-services – Emergence of Micro-Service Architecture – Design Patterns of Micro-services – The Mini Web Service Architecture

Defining Microservices

Definition

Microservices is an architectural style in which a large application is divided into a collection of small, independent services. Each service performs a specific business function and communicates with other services through APIs.

Features of Microservices

- Small and independent services.
- Loosely coupled architecture.
- Independently deployable.
- Technology independent.
- Highly scalable and flexible.

Advantages

- Faster development and deployment.
- Easy maintenance.
- Better fault isolation.
- Improved scalability.
- Supports continuous delivery.

Example

In an online shopping application:

- User Service manages users.
- Product Service manages products.
- Order Service manages orders.
- Payment Service manages payments.

Each service works independently and communicates through APIs.

Emergence of Microservice Architecture

Introduction

Before microservices, most applications followed a **Monolithic Architecture**, where all functionalities were built into a single application.

Problems with Monolithic Architecture

- Difficult to maintain large applications.
- Scaling requires scaling the entire application.
- Slow deployment process.
- Failure in one module may affect the entire system.
- Difficult for large teams to work simultaneously.

Why Microservices Emerged

- Need for better scalability.
- Faster software delivery.
- Cloud computing growth.
- Agile and DevOps practices.
- Independent development and deployment.

Evolution

Traditional Systems
↓
Monolithic Architecture
↓
Service-Oriented Architecture (SOA)
↓
Microservice Architecture

Benefits of Emergence

- Rapid development.
- Better performance.
- Easy updates.
- Independent service management.

Design Patterns of Microservices

Design patterns provide solutions to common problems in microservice-based applications.

1. API Gateway Pattern

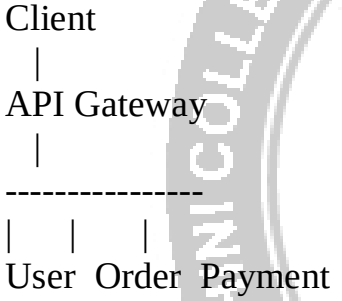
Definition

API Gateway acts as a single entry point for all client requests.

Functions

- Request routing.
- Authentication and authorization.
- Load balancing.
- Security management.

Diagram



Advantages

- Simplifies client communication.
- Improves security.
- Reduces complexity.

---

2. Database per Service Pattern

Definition

Each microservice maintains its own database.

Advantages

- Independent data management.
- Better scalability.
- Improved security.
- Reduced coupling.

Diagram

User Service → User DB
Order Service → Order DB
Payment Service → Payment DB

---

3. Service Discovery Pattern

Definition

Allows services to automatically locate and communicate with each other.

Advantages

- Dynamic service registration.
- Improved reliability.
- Easy scalability.

Example

When Order Service needs Payment Service, it finds the service location automatically.

---

4. Circuit Breaker Pattern

Definition

Prevents a service from repeatedly calling a failed service.

Working

- Detects failures.
- Stops requests temporarily.
- Recovers automatically after some time.

Advantages

- Prevents system crashes.
- Improves fault tolerance.
- Enhances reliability.

The Mini Web Service Architecture

Definition

Mini Web Service Architecture is a simple architecture that demonstrates how clients, APIs, services, and databases interact in a microservice environment.

Components

1. Client Layer

- Web browsers.
- Mobile applications.
- Desktop applications.

Functions

- Sends requests.
- Receives responses.

---

2. API Gateway Layer

Functions

- Accepts client requests.
- Routes requests to appropriate services.
- Handles security and authentication.

---

3. Service Layer

Contains

- User Service.
- Product Service.
- Order Service.
- Payment Service.

Functions

- Business logic processing.
- Data validation.
- Service communication.

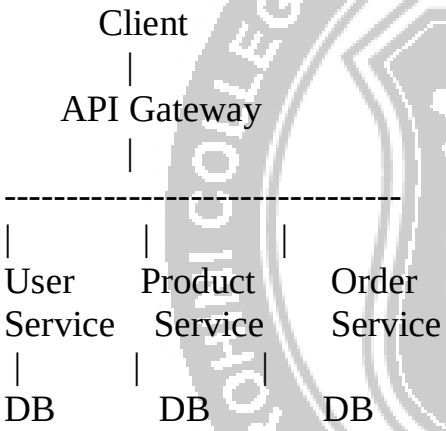
---

4. Database Layer

Functions

- Stores application data.
- Provides data access.
- Maintains data consistency.

Mini Web Service Architecture Diagram



Working of Mini Web Service Architecture

Step 1

Client sends a request.

Step 2

API Gateway receives the request.

Step 3

API Gateway forwards the request to the appropriate microservice.

Step 4

Microservice processes the request.

Step 5

Microservice accesses the database.

Step 6

Response is sent back to the client.

Conclusion

- Microservices divide applications into small independent services.
- They emerged to overcome the limitations of monolithic systems.
- Design patterns such as API Gateway, Database per Service, Service Discovery, and Circuit Breaker improve efficiency and reliability.
- Mini Web Service Architecture consists of Client Layer, API Gateway, Service Layer, and Database Layer working together to provide scalable and maintainable applications.

1. Defining Microservices

Introduction

Microservices is an architectural style that structures an application as a collection of small, loosely coupled, independently deployable services. Each service focuses on a single business capability and communicates with other services using lightweight protocols such as HTTP/REST APIs.

The microservice architecture was introduced to overcome the limitations of traditional monolithic applications and to support modern cloud-based systems.

Definition

According to industry standards,

"Microservices are small autonomous services that work together to build a larger application."

Each microservice:

- Has its own business logic.
- Runs independently.
- Has its own database.
- Can be developed by different teams.
- Can be deployed without affecting other services.

Characteristics of Microservices

1. Single Responsibility

Each service performs only one business function.

Example:

- User Service
- Product Service
- Payment Service
- Order Service

2. Loose Coupling

Services are independent and communicate through APIs.

Advantages:

- Easy maintenance
- Better flexibility
- Reduced dependency

3. Independent Deployment

Each service can be updated and deployed separately.

Benefits:

- Faster release cycles
- Minimal downtime
- Easy bug fixing

4. Scalability

Only the required service is scaled.

Example:
If payment traffic increases, only the Payment Service is scaled instead of the whole application.

5. Fault Isolation

Failure of one service does not crash the entire application.

Example:
If Notification Service fails, Order Service continues functioning.

Advantages of Microservices

Technical Advantages

- High scalability
- Better fault tolerance
- Independent deployment
- Faster development
- Technology diversity

Business Advantages

- Faster time to market
- Reduced operational cost
- Improved customer satisfaction
- Continuous delivery support

Disadvantages of Microservices

- Complex architecture
- Network overhead
- Difficult testing
- Security challenges
- Data consistency issues

2. Emergence of Microservice Architecture

Monolithic Architecture

Initially, software applications were built using Monolithic Architecture.

Structure

Application
- User Module
- Product Module
- Payment Module
- Order Module

All modules are packed and deployed together.

Problems with Monolithic Architecture

1. Difficult Maintenance

As application size increases, code becomes difficult to manage.

2. Scaling Problems

Entire application must be scaled even when only one module needs more resources.

3. Deployment Issues

A small update requires redeployment of the entire application.

4. Technology Limitation

Entire application uses the same technology stack.

5. System Failure

Failure in one module can affect the entire application.

Evolution Toward Microservices

Traditional Systems
↓
Monolithic Architecture
↓
Service-Oriented Architecture (SOA)
↓
Microservice Architecture

Reasons for Emergence

Cloud Computing Growth

Cloud platforms support distributed deployment.

Agile Development

Frequent software releases require flexible architecture.

DevOps Culture

Continuous integration and deployment need independent services.

Big Data Applications

Large-scale applications require scalable architecture.

Business Agility

Organizations need faster feature delivery.

---

3. Design Patterns of Microservices

Design patterns provide standardized solutions for common architectural problems.

---

1. API Gateway Pattern

Definition

API Gateway acts as a single entry point for all client requests.

Functions

- Request routing
- Authentication
- Authorization
- Rate limiting
- Load balancing
- Logging

Architecture

Client
|
API Gateway
|

| | |
User Order Payment

Advantages

- Simplifies communication
- Enhances security
- Reduces client complexity

---

2. Database Per Service Pattern

Definition

Every microservice owns and manages its own database.

Structure

User Service
|
User DB

Order Service
|
Order DB

Payment Service
|
Payment DB

Advantages

- Independent data management
- Better scalability
- Improved security

---

3. Service Discovery Pattern

Definition

Services automatically discover and communicate with each other.

Types

Client-Side Discovery

Client finds the service location.

Server-Side Discovery

Load balancer finds the service.

Benefits

- Dynamic routing
- Easy scalability
- Reduced manual configuration

---

4. Circuit Breaker Pattern

Definition

Prevents repeated calls to a failed service.

States

Closed State

Requests flow normally.

Open State

Requests are blocked.

Half-Open State

Limited requests are allowed for testing.

Benefits

- Prevents cascading failures
- Improves reliability
- Enhances fault tolerance

---

5. Strangler Pattern

Definition

Used to gradually migrate a monolithic application into microservices.

Benefits

- Reduced risk
- Smooth migration
- Incremental modernization

---

4. Mini Web Service Architecture

Definition

Mini Web Service Architecture is a simplified architecture illustrating how microservices interact through APIs and databases.

Components

1. Client Layer

Users interact through:

- Web browsers
- Mobile applications
- Desktop applications

Responsibilities:

- Send requests
- Receive responses

---

2. API Gateway Layer

Responsibilities:

- Request routing
- Authentication
- Security
- Traffic management

---

3. Microservice Layer

Contains:

User Service

Manages user accounts.

Product Service

Manages products.

Order Service

Handles orders.

Payment Service

Processes payments.

Notification Service

Sends notifications.

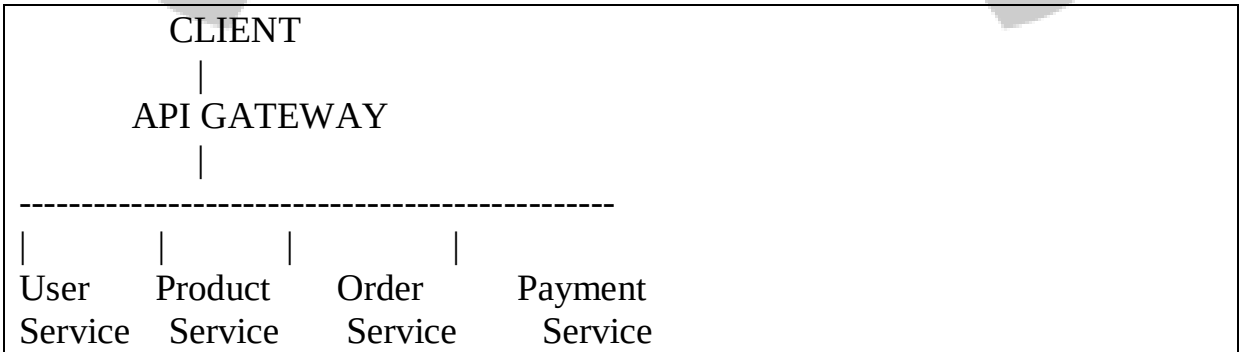
---

4. Database Layer

Each service maintains its own database.

Responsibilities:

- Data storage
- Data retrieval
- Backup management
- **Architecture Diagram**



User DB	Product DB	Order DB	Payment DB

Working of Mini Web Service Architecture

Step 1

Client sends request.

Step 2

API Gateway receives request.

Step 3

Authentication and authorization are verified.

Step 4

Gateway forwards request to appropriate microservice.

Step 5

Microservice processes business logic.

Step 6

Database operations are performed.

Step 7

Response is returned to API Gateway.

Step 8

API Gateway sends response to client.

Comparison: Monolithic vs Microservices

Monolithic	Microservices
Single application	Multiple services

Monolithic	Microservices
Single deployment	Independent deployment
Shared database	Separate databases
Difficult scaling	Easy scaling
Less fault tolerant	High fault tolerance
Slow updates	Faster updates