Microservices - Flask RESTFul

Definition of Microservices

- Microservices is an **architectural style** where applications are built as a **collection of small, independent, loosely-coupled services**.
- Each service handles a **single business capability** (e.g., Login Service, Order Service).
- Services communicate using **lightweight protocols** like HTTP/REST, gRPC, or messaging queues.

Why Microservices?

- Faster development and deployment
- Independent scaling
- Technology freedom (Java + Python + Node.js in same project)
- Better maintainability
- Fault isolation (one service fails → system still runs)
- Ideal for cloud platforms like AWS, Azure, GCP, Docker & Kubernetes

Microservices vs Monolithic Architecture

Monolithic	Microservices
Single large codebase	Many small services
Harder to scale	Each service scales independently
One tech stack	Polyglot (multiple languages)
One failure → full downtime	Service isolation
Slow deployment	Faster CI/CD

Microservices Characteristics & Principles

Key Characteristics

1. **Autonomous services**
2. **Decentralized data**
3. **Lightweight communication**
4. **Resilience & fault tolerance**
5. **Independent deployment**
6. **DevOps culture**
7. **Continuous delivery**

Microservices Principles (12-factor inspired)

1. **Single Responsibility Principle**
2. **API-first design**
3. **Loose coupling, high cohesion**
4. **Decentralized governance**
5. **Decentralized data management**
6. **Infrastructure automation (CI/CD)**
7. **Observability: logging, monitoring, tracing**
8. **Scalability & isolation**
9. **Security & API gateway patterns**

Microservices Communication

- **REST APIs (HTTP/JSON)**
- **Message queues:** RabbitMQ, Kafka
- **gRPC**
- **Service Registry:** Consul, Eureka
- **API Gateway:** Kong, Nginx, AWS API Gateway

Introducing Flask-RESTful for Microservices

What is Flask-RESTful?

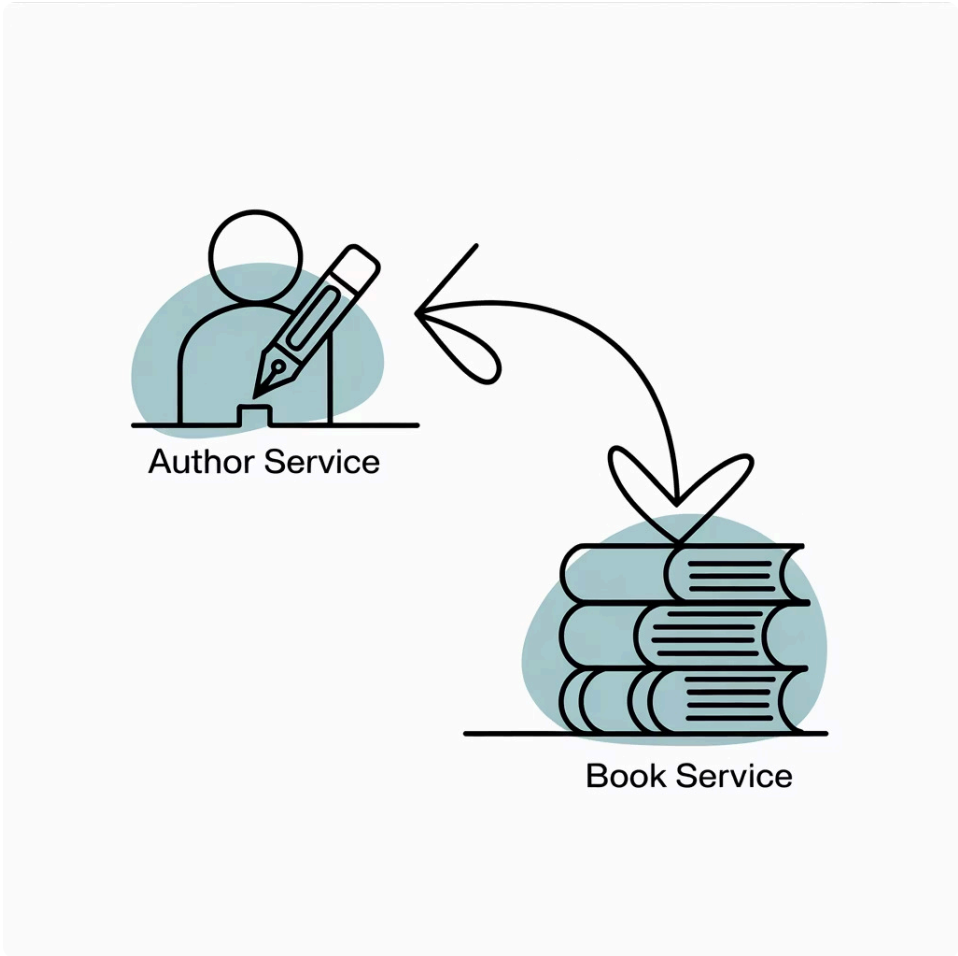
- Flask-RESTful is a Flask extension that makes it **easy to build REST APIs**.
- Works perfectly for **small & lightweight microservices**.
- Helps in rapid prototyping and production-ready services.

Why Flask-RESTful for Microservices?

- Simple and minimal
- Easy to scale horizontally (with Docker)
- Lightweight and fast
- Clear separation of routes and logic
- Built-in request parsing
- Easy error handling
- Supports JSON by default
- Can integrate with SQLAlchemy, JWT, Marshmallow, Swagger

Typical Microservices Built with Flask

- Product Service
- Order Service
- Login/Auth Service
- Notification Service
- Payment Service
- Inventory Service



Microservice Example

In this example, there are two services viz., Book Service, which will serve all the functionalities related to book and Author Service, has functionalities related to author. If we send request to author service to fetch the books for the given genre, then it will connect to Book Service to fetch the books as author service don't have any information related to books.