Microservices:

Transactions always not consistency in different services and databases.

So need some design patterns:

Saga pattern features:

1. Ensure Transaction Consistency in a distributed system.
2. Avoid two-phase commit(2PC) which lead performance bottlenecks
3. Fallback when error.

*A saga:*

*Each transaction update database and publish event to trigger next step in workflow.*

*If any step fails, the previous steps must be rolled back.*

Types of saga pattern:

1. Event driven approach(**Choreography-Based Saga**)

* Each service use event broker(kafka) to produce and listen for control workflow, no central orchestrator controls.

- Any step fail, it will fallback.

1. Orchestration-based saga(Centralized control)

- An Orchestrator class manage all workflow.

- Any step fail, it will fallback.

Take an e-commerce application as example:

Order service

Payment service

Inventory service - Reserves items

Shipping service