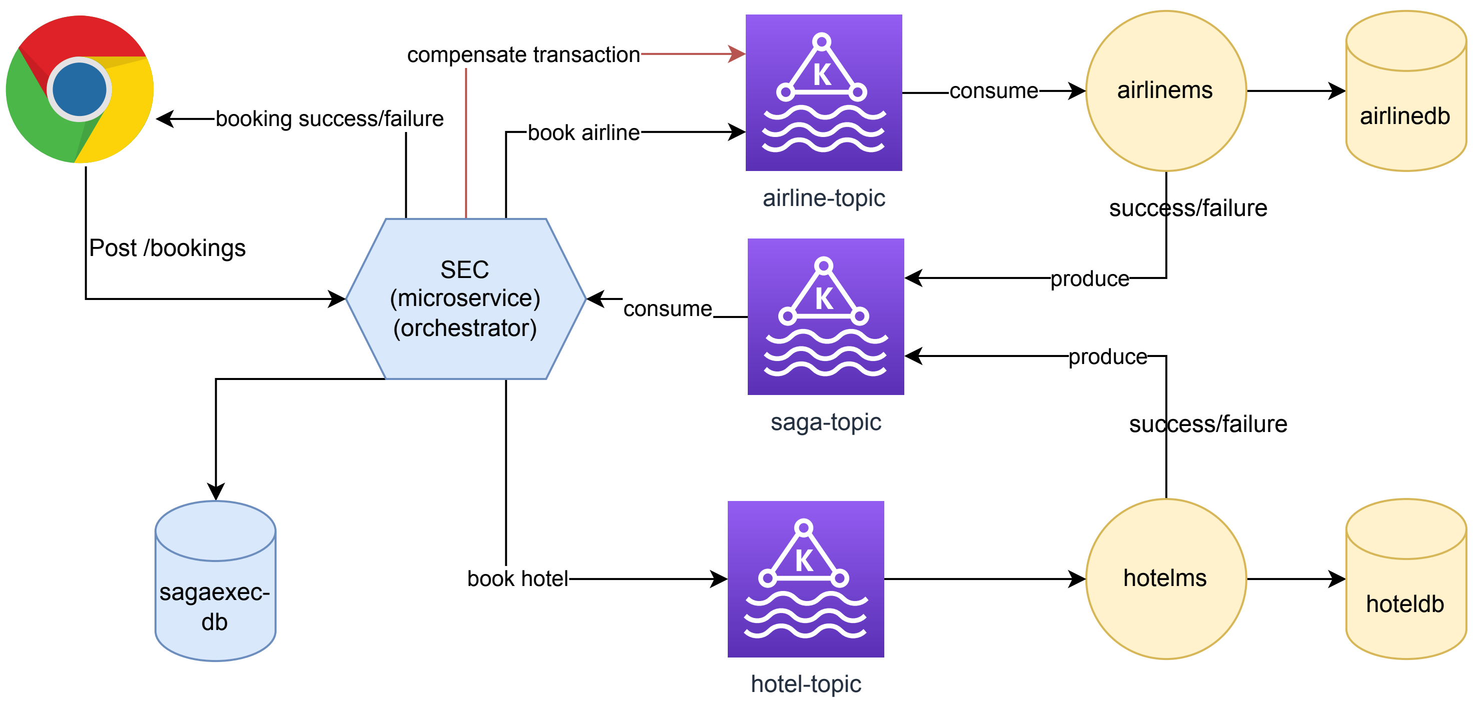
Transaction Management
(Saga Pattern)
Eventual Consistent

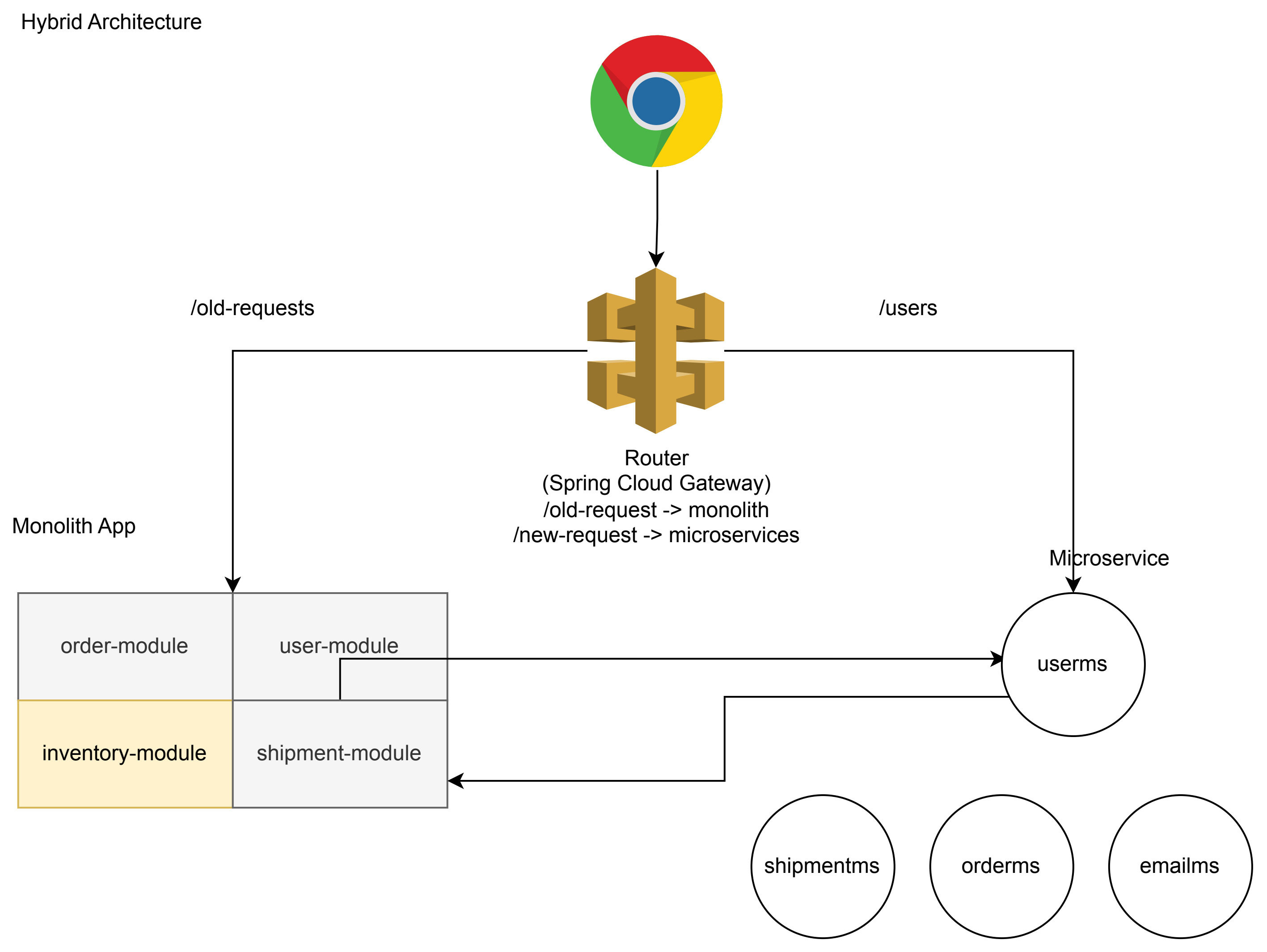
Use Cases:

1. Both airlinems and hotelms successfully update the database
2. airlinems fails, terminate the transaction
3. airlinems succeeds but hotelms fails to update DB -> revert airline booking
4. Both airlinems and hotelms fail to update the Database

SEC: Saga Execution Coordinator



Strangler Vine
(Decomposition Strategy)



Microervices Design Patterns

Integration Patterns

API Gateway
Aggregator
Chained
Branch

Database Patterns

Database Per Servcice
Shared Database
Saga
CQRS
Event Sourcing

Cross Cutting Patterns

Service Discovery
Service Registry
Client-side Load Balancing
External Configuration
Circuit Breaker

Decomposition Patterns

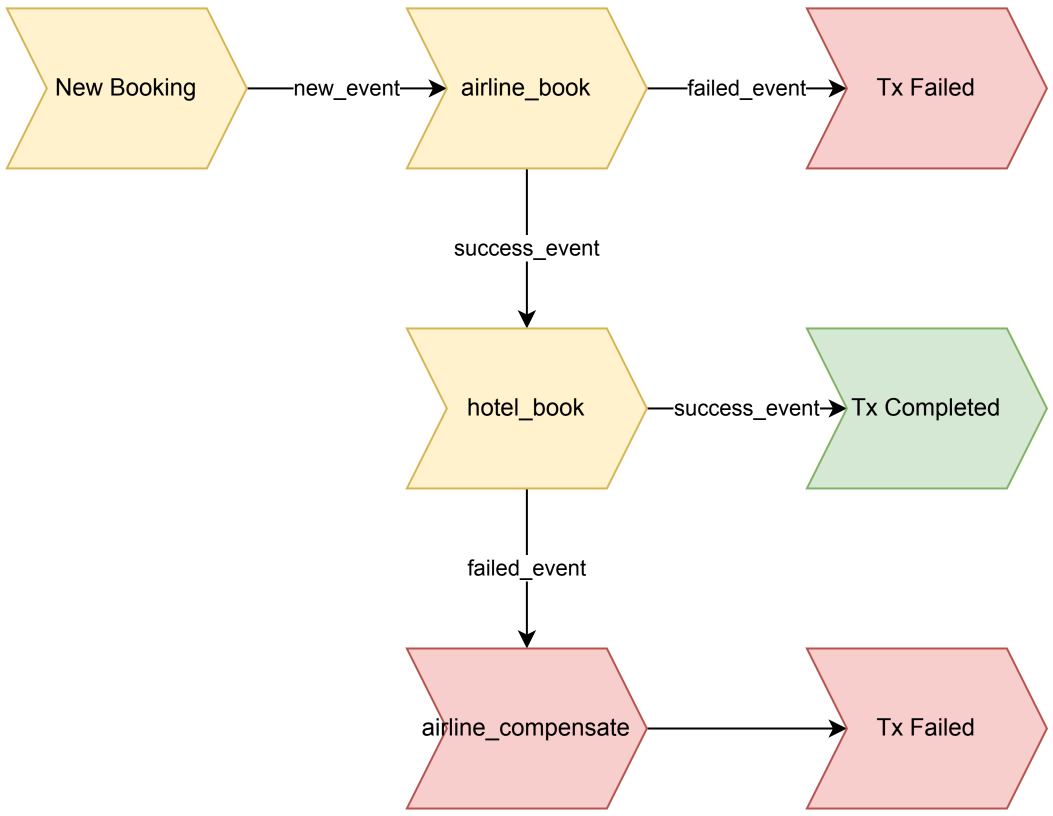
Strangle Vine

Observability Patterns

Distributed Tracing
Log Aggregation
Metrics
Health Check

Finite State Machine
(used in saga(sec) implementation)

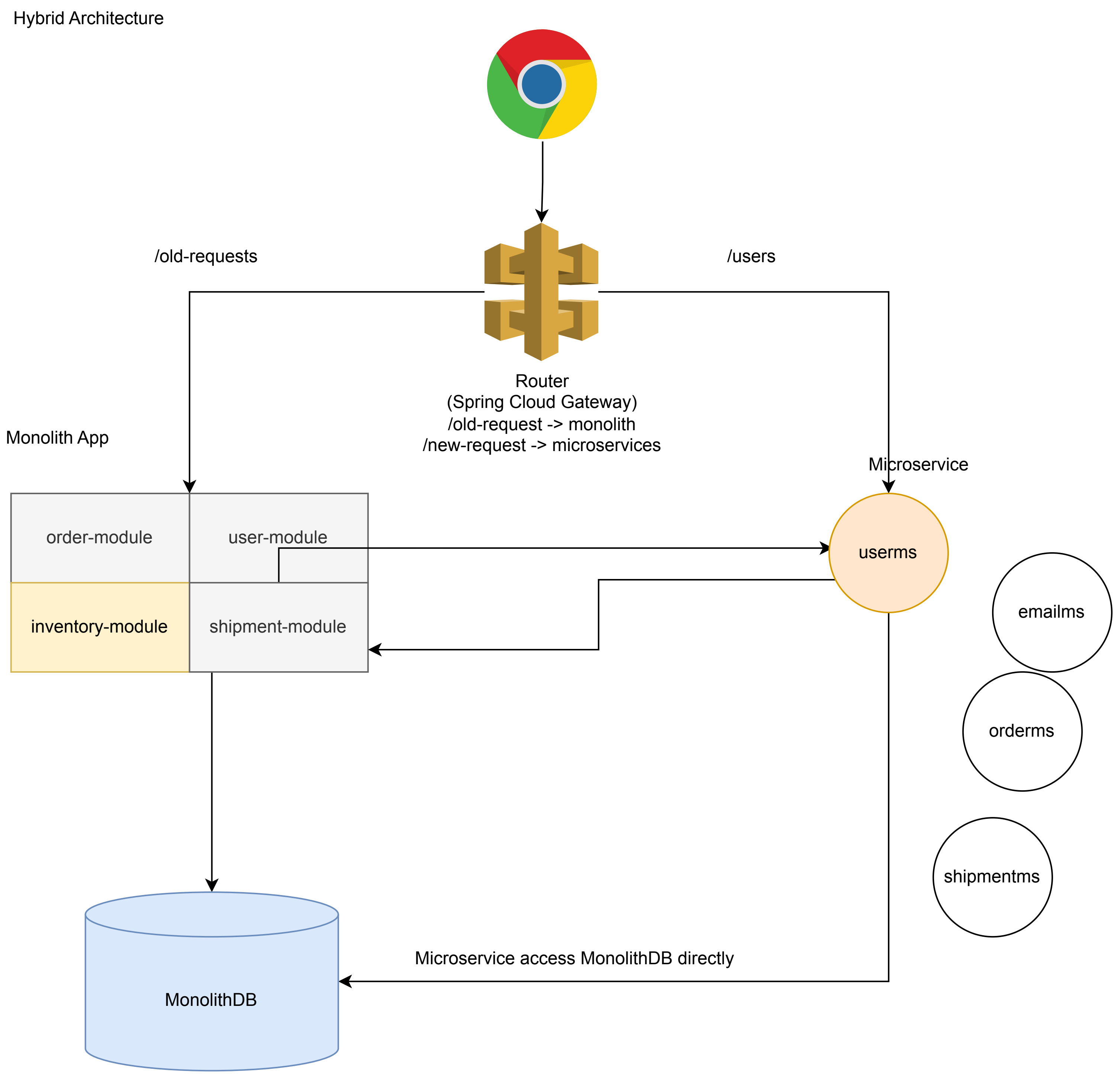
States and Transitions:



Database with Hybrid Architecture

Technique# 1 Monolith and Microservices both access MonolithDB directly

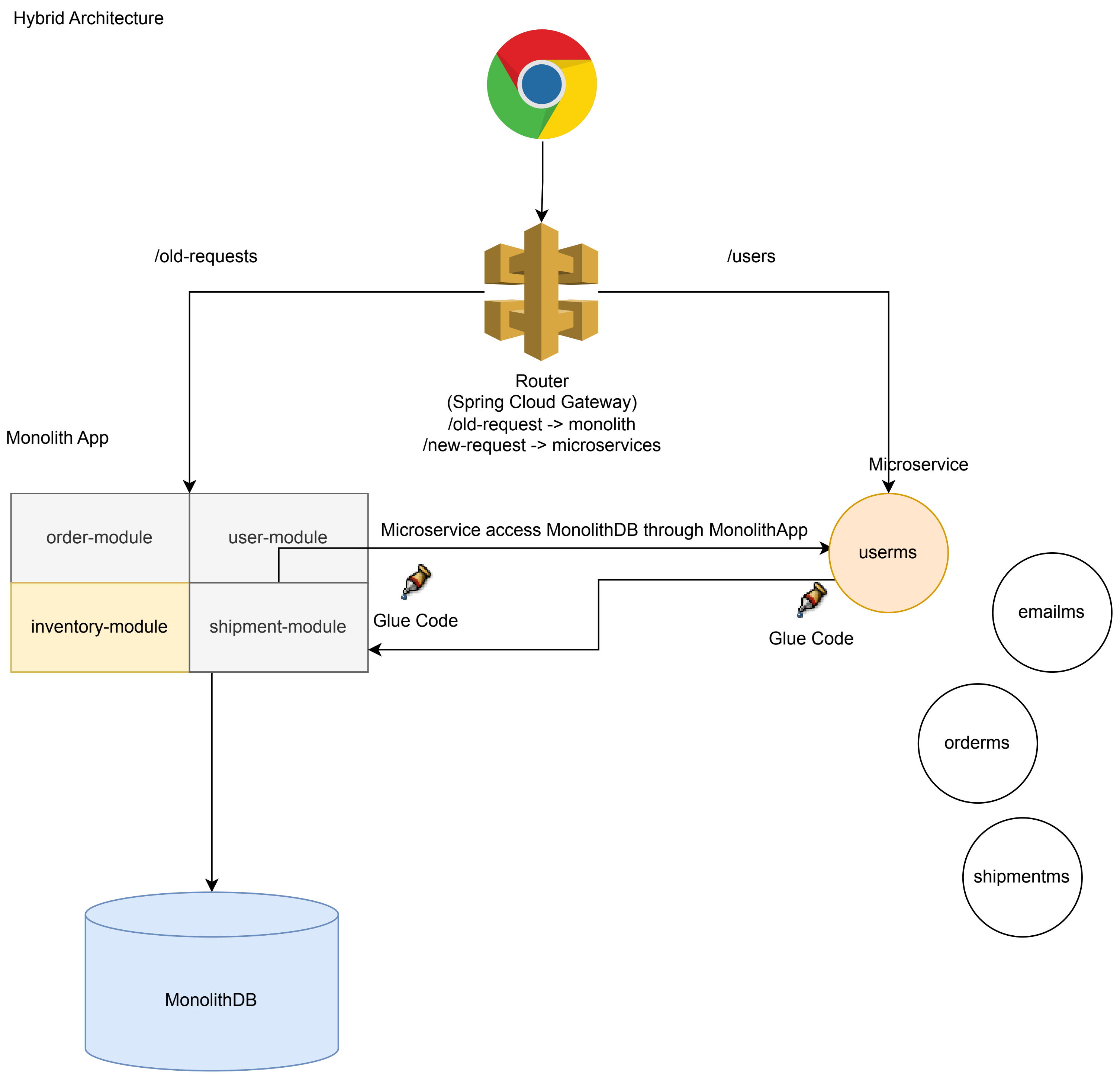
Strangler Vine
(Decomposition Strategy)



Database with Hybrid Architecture

Technique# 1 Microservices access MonolithDB through MonolithApp

Strangler Vine
(Decomposition Strategy)

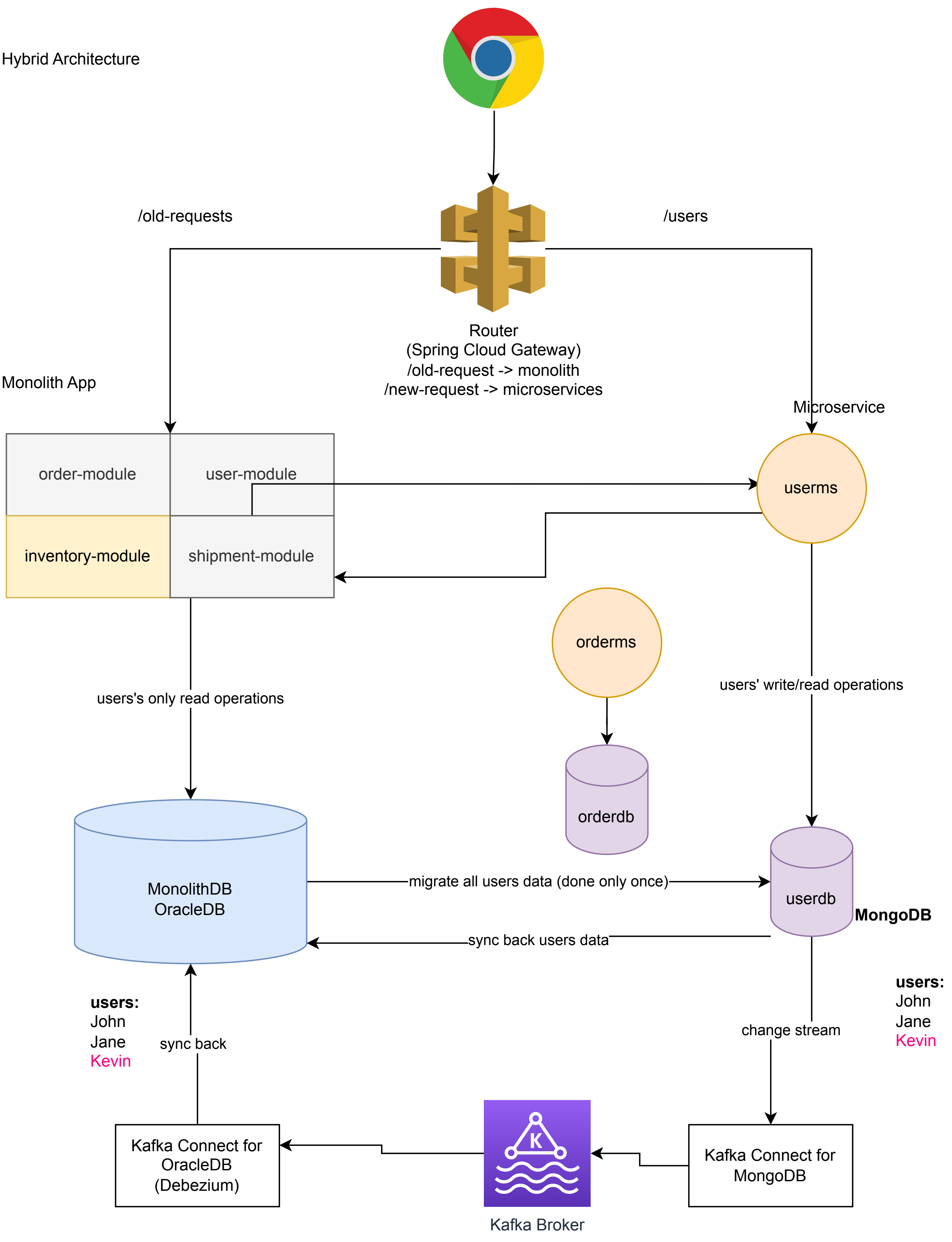


Database with Hybrid Architecture

Technique# 1 Microservices access MicroserviceDB
MonolithApp access MonolithDB
MonolithDB and MicroservicesDB need to be in sync(if required)

Strangler Vine
(Decomposition Strategy)

Hybrid Architecture



/old-requests
/users
Router
(Spring Cloud Gateway)
/old-request -> monolith
/new-request -> microservices
Monolith App
Microservice
order-module
user-module
inventory-module
shipment-module
userms
orderms
orderdb
users's only read operations
users' write/read operations
MonolithDB
OracleDB
migrate all users data (done only once)
userdb
MongoDB
sync back users data
users:
John
Jane
Kevin
sync back
change stream
users:
John
Jane
Kevin
Kafka Connect for OracleDB (Debezium)
Kafka Broker
Kafka Connect for MongoDB