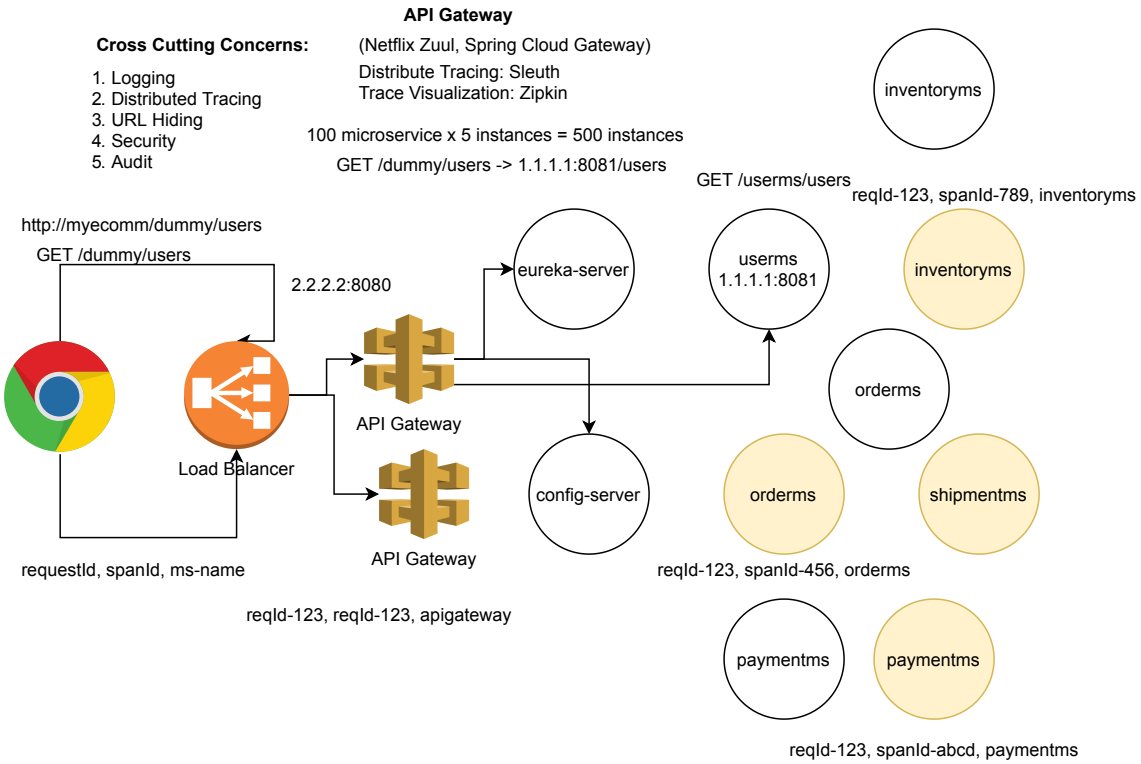
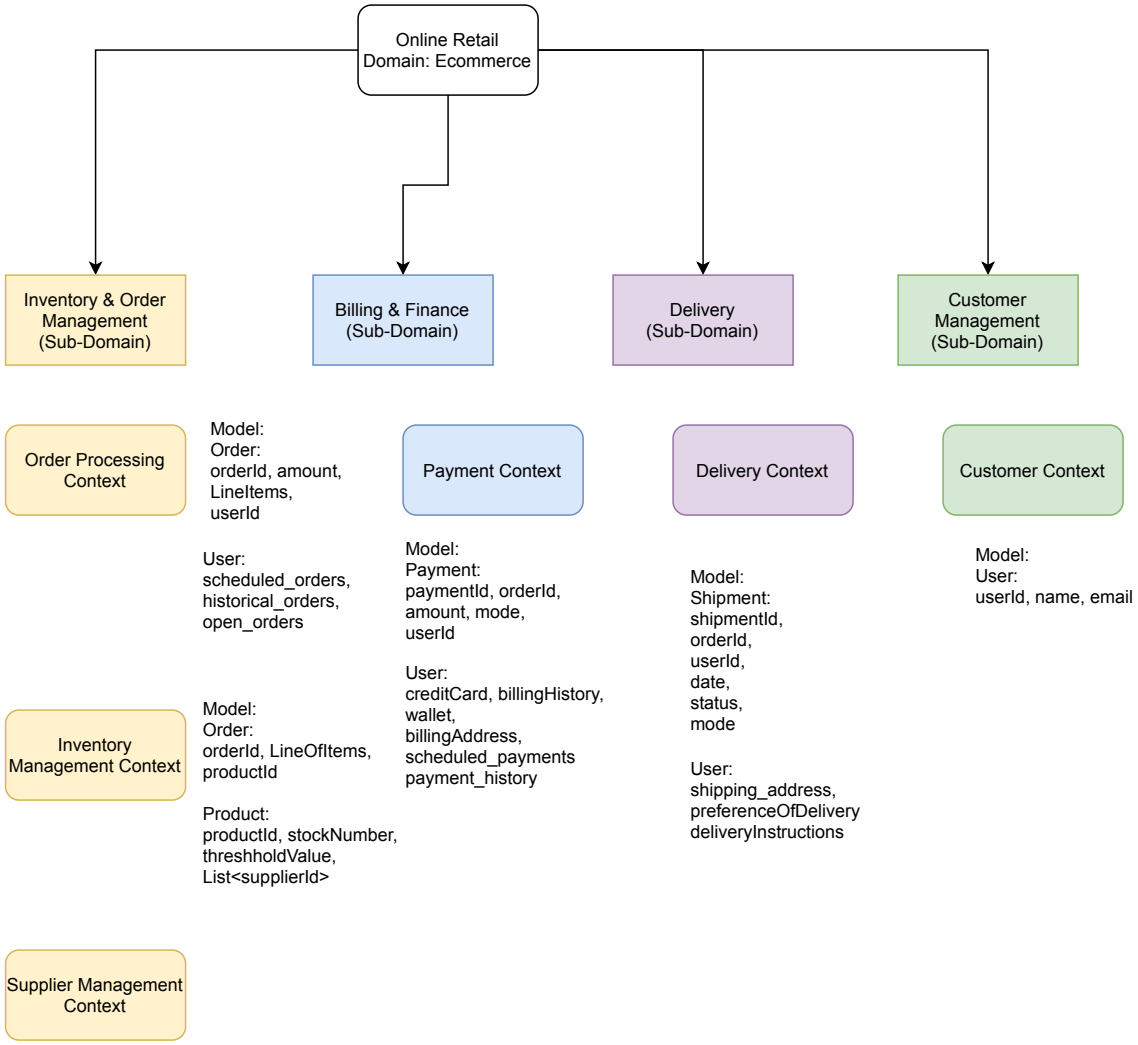

API Gateway
Cross Cutting Concerns:
1. Logging
2. Distributed Tracing
3. URL Hiding
4. Security
5. Audit
(Netflix Zuul, Spring Cloud Gateway)
Distribute Tracing: Sleuth
Trace Visualization: Zipkin
100 microservice x 5 instances = 500 instances
GET /dummy/users -> 1.1.1.1:8081/users
inventoryms
GET /userms/users
reqId-123, spanId-789, inventoryms
http://myecomm/dummy/users
GET /dummy/users
2.2.2.2:8080
eureka-server
userms
1.1.1.1:8081
inventoryms
orderms
API Gateway
Load Balancer
config-server
orderms
shipmentms
API Gateway
requestId, spanId, ms-name
reqId-123, spanId-456, orderms
reqId-123, reqId-123, apigateway
paymentms
paymentms
reqId-123, spanId-abcd, paymentms

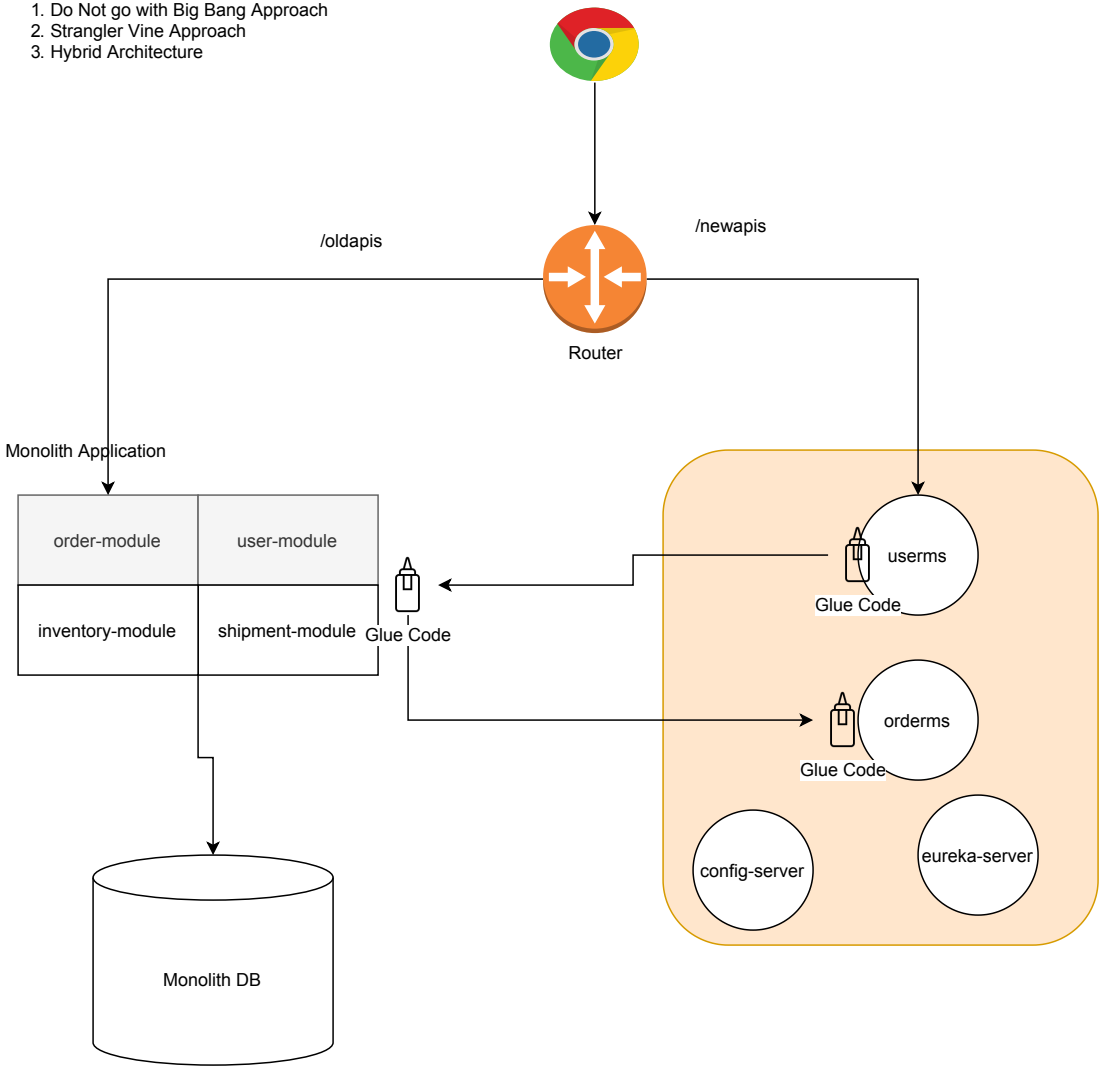
Domain Driven Design

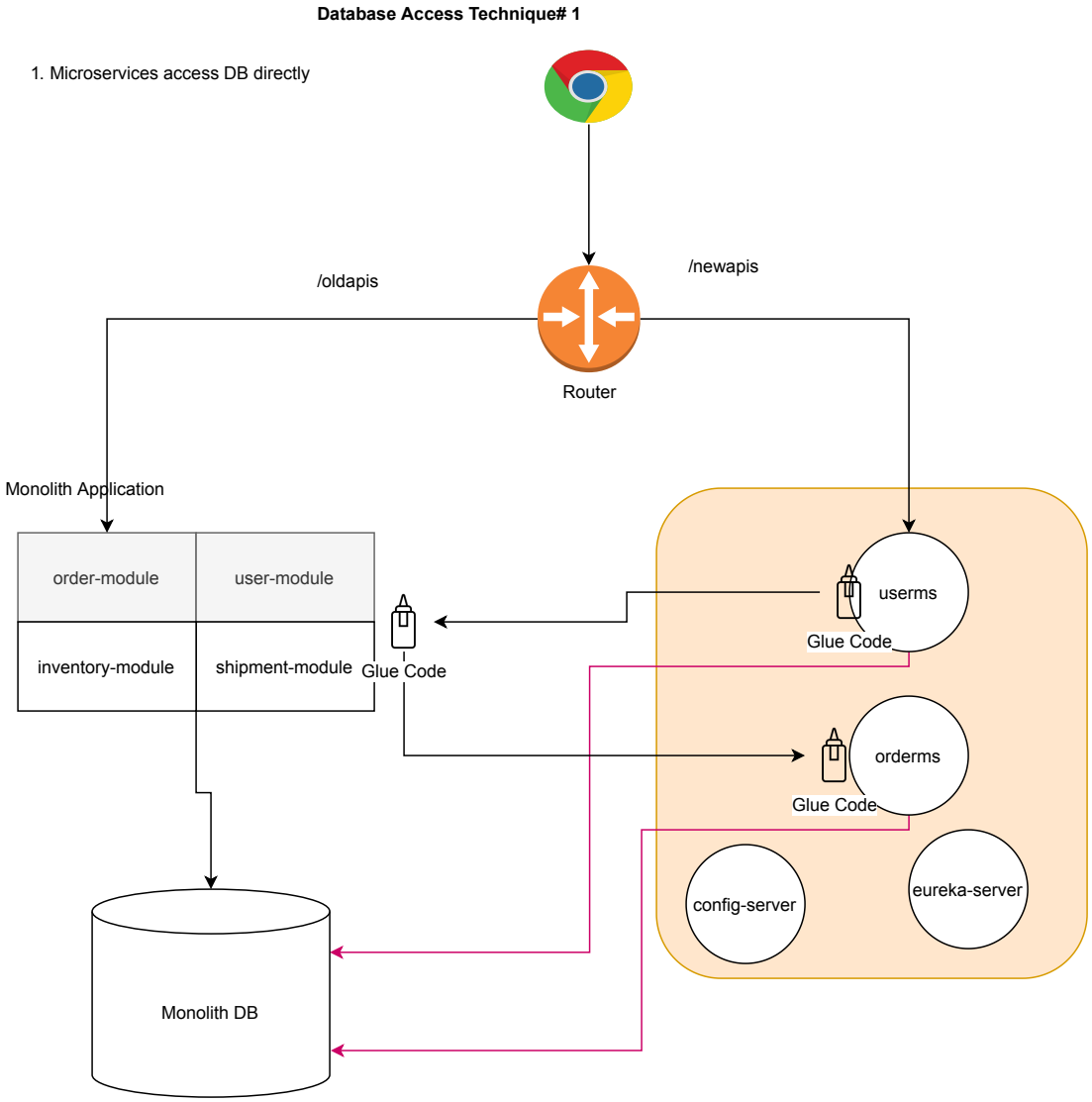
1. Domain/Subdomain
2. Context/Bounded Context
3. Model
4. Ubiquitous Langauge

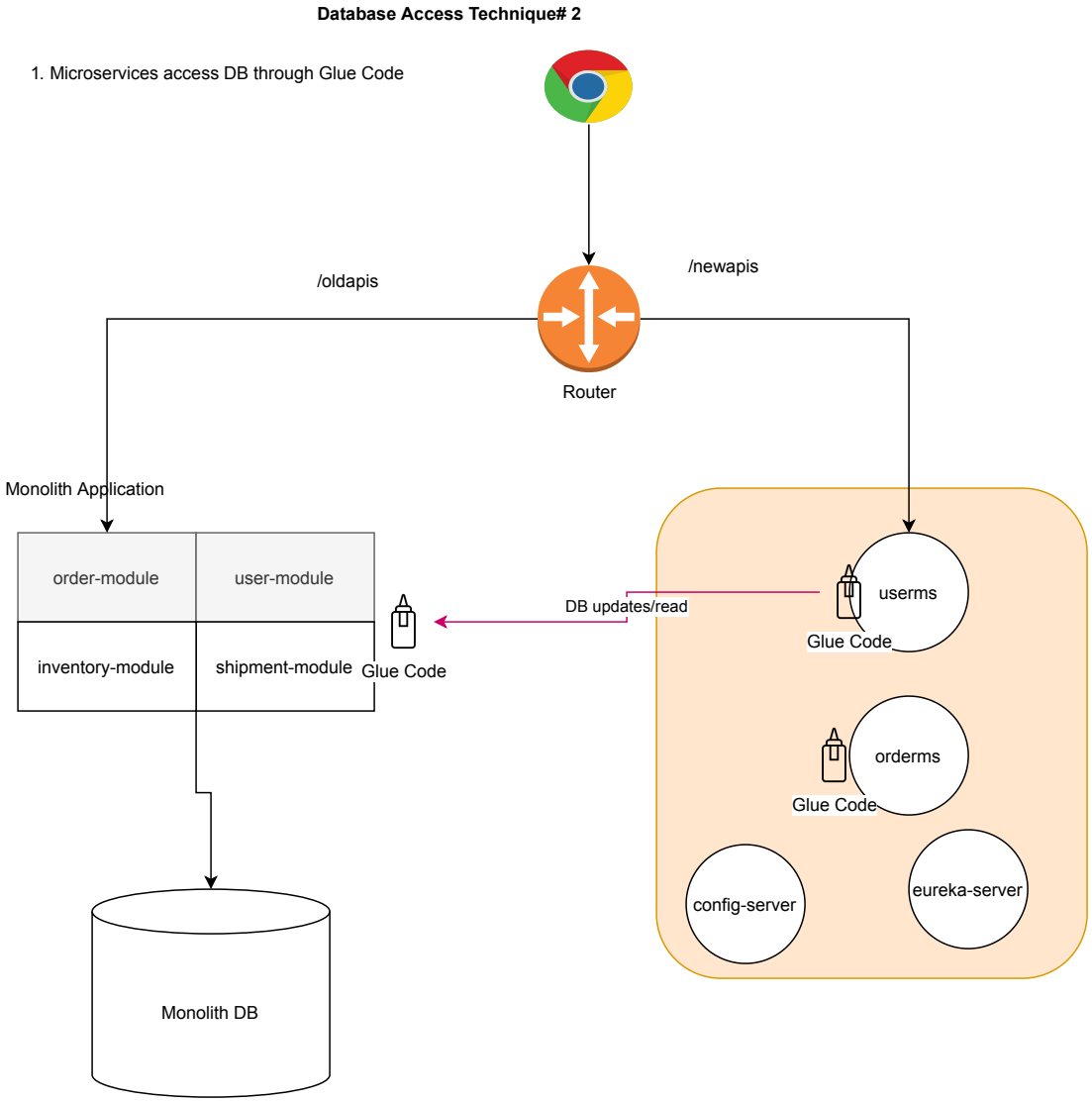


Migration from Monolith to Microservices

1. Do Not go with Big Bang Approach
2. Strangler Vine Approach
3. Hybrid Architecture

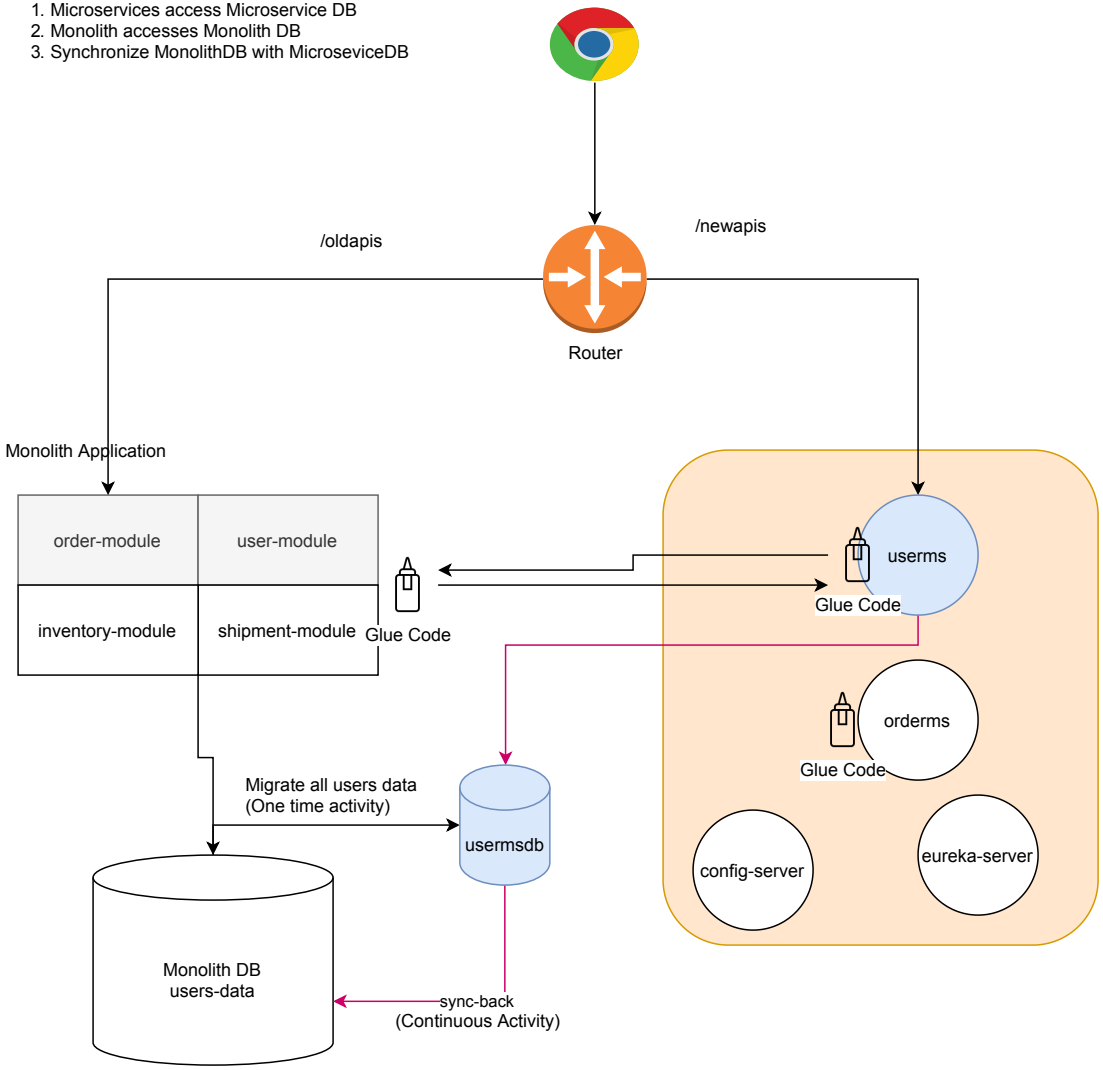


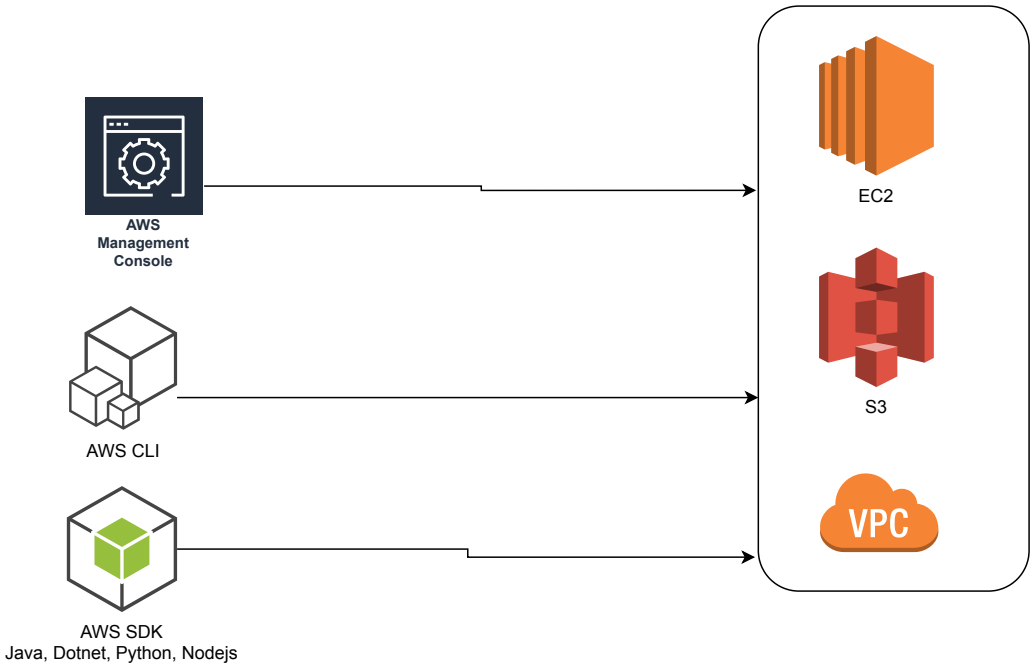

Database Access Technique# 1
1. Microservices access DB directly
/oldapis
/newapis
Router
Monolith Application
order-module
user-module
inventory-module
shipment-module
Glue Code
userms
Glue Code
orderms
Glue Code
config-server
eureka-server
Monolith DB

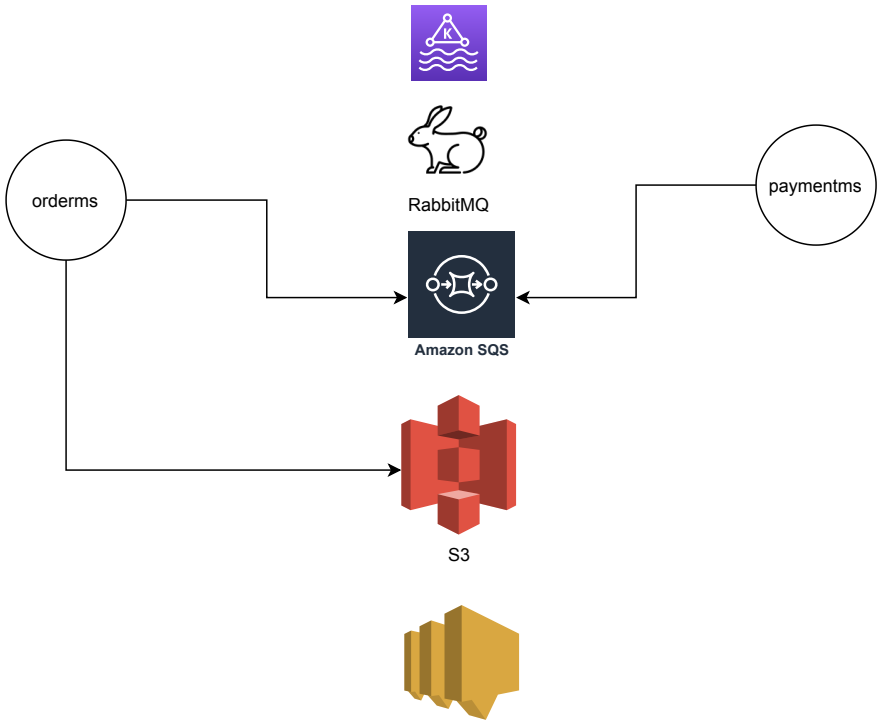

Database Access Technique# 2
1. Microservices access DB through Glue Code
/oldapis
/newapis
Router
Monolith Application
order-module
user-module
inventory-module
shipment-module
Glue Code
DB updates/read
userms
Glue Code
orderms
Glue Code
config-server
eureka-server
Monolith DB

Database Access Technique# 3

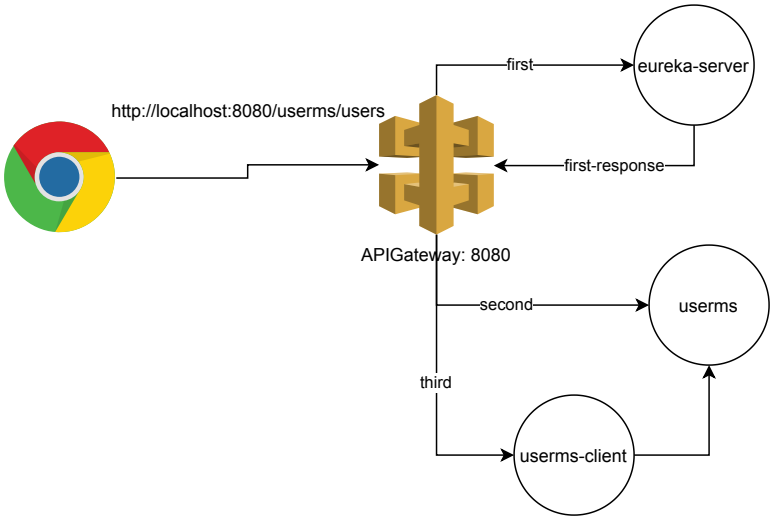
1. Microservices access Microservice DB
2. Monolith accesses Monolith DB
3. Synchronize MonolithDB with MicroseviceDB

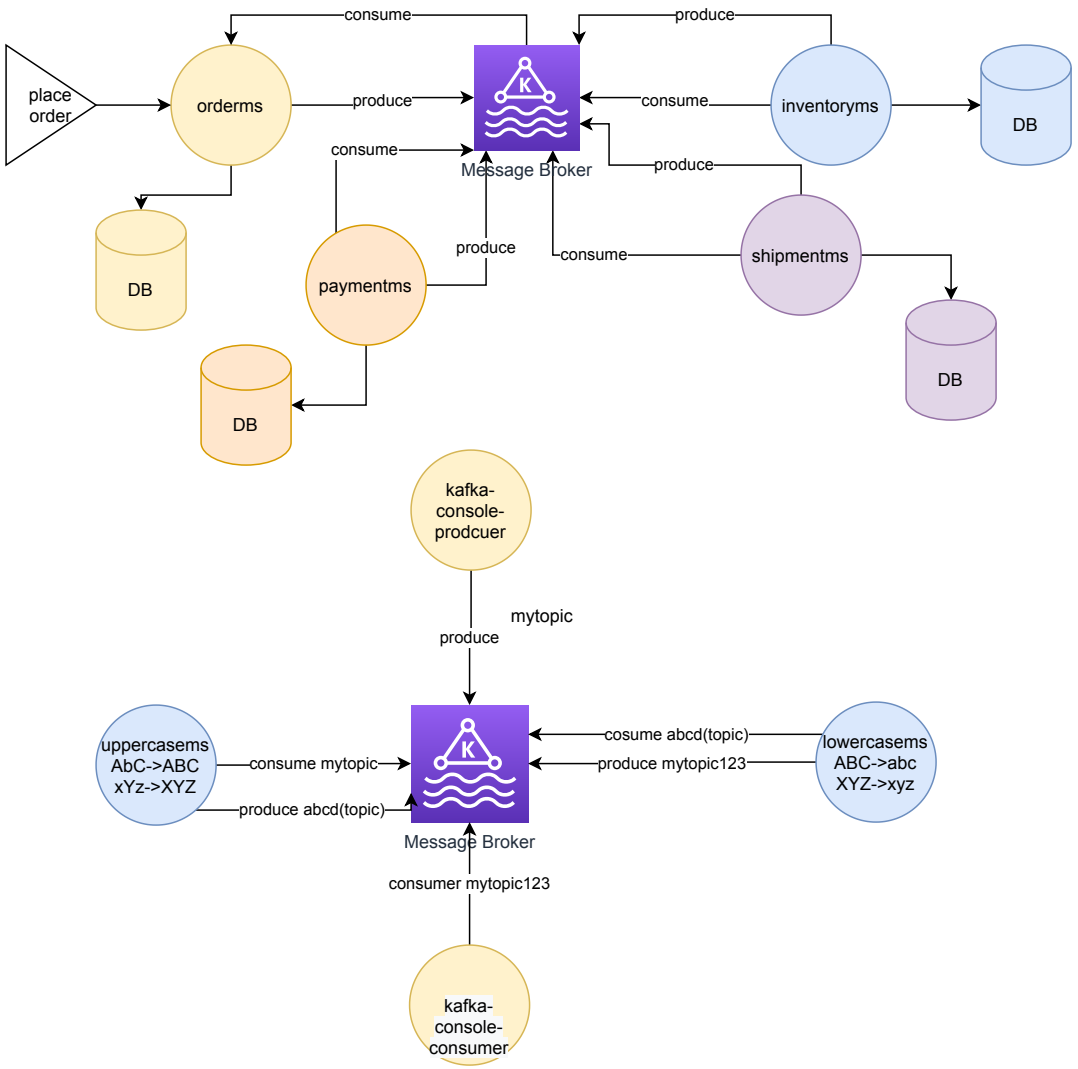



AWS
Management
Console
AWS CLI
AWS SDK
Java, Dotnet, Python, Nodejs
EC2
S3
VPC


orderms
RabbitMQ
paymentms
Amazon SQS
S3

SNS


first
eureka-server
http://localhost:8080/userms/users
first-response
APIGateway: 8080
second
userms
third
userms-client

Spring Cloud Stream

(Event Driven Arhitecture)