SHOPPING PORTAL PROJECT

MICROSERVICES ARCHITECTURE

Hedi Abed

15/30/2020

AGENDA

1. MONOLITHIC APPLICATIONS
2. MICROSERVICES APPLICATIONS
3. SHOPPING PORTAL ARCHITECTURE

MONOLITHIC APPLICATIONS

REMINDS

MONOLITHIC

Architecture

Monolithic literally means a single block of stone and in software architecture, it is more referring to a one single application.

Single Application:

- One Code Base
- One Build System
- Single executional program (ie WAR or EAR file)

MONOLITHIC

Traits

- Code is stored together
- We use one database
- Releases as one big version
- Scaling is an all or nothing situation

MONOLITHIC

Benefits

- Development is easy
- Deployment is easy
- Testing is simplified

MONOLITHIC

Problems

- Business requirements growth = more complexity
- Difficult to modify
- Technology Lock In
- CI/CD difficult

MICROSERVICES APPLICATIONS

LET'S DIVE IN

MICROSERVICES

Architecture

Microservices are small targeted services that structures an application while each service has its own repository

These services:

Loosely coupled

Independently deployable

Highly maintainable

MICROSERVICES

Architecture

With a Microservice Architecture:

- Applications are composed using individual microservices
- Each service will typically have its own database
- Each microservice is independently deployable
- Scaling of individual services is much more possible
- CI/CD becomes easier since services are smaller and less complex to deploy

MICROSERVICES

Benefits

- Easy to understand & develop
- Software Quality
- Scalability
- Reliability
- Technology flexibility

MICROSERVICES

Cons

- Integration testing can be difficult
- Deployments are more complex
- Operational cost with each service
- Additional hardware resources

MICROSERVICES

Decomposing

Decomposing is the process of taking a larger monolithic application and breaking it up into microservices and it’s more of an ‘art’ than a science.

MICROSERVICES

Decomposing

Strategies we can use:

- By Business Capability
- By Domain Objects
- By action verbs
- By Nouns

MICROSERVICES

Single Responsibility Principle

- SRP is a term coined about object oriented programming.
- SRP says a class should have just one reason to change and that the meaning of classes should be very specific in what they do.
- SRP can also be applied to microservices

SHOPPING PORTAL

LOOKING AHEAD

MICROSERVICES

MICROSERVICES

Provides user account operation functionality and attached to SQL database.
Ex: CRUD, Activate, Deactivate and Verified ...

MICROSERVICES

Provides product operation functionality and attached to SQL database.
Ex: CRUD, status ...

MICROSERVICES

Provides product inventory operation functionality and attached to NoSQL database.
Ex: retrieve, increase, decrease and infinity.

MICROSERVICES

Provides shipping operation functionality and attached to NoSQL database.
Ex: CRU

MICROSERVICES

Provides payment functionality and attached to online payment services.
Ex: Stripe, PayPal ...

MICROSERVICES

Provides personal shop operation functionality and attached to NoSQL database.
Ex: CRUD ...

MICROSERVICES

Provides personal shop operation functionality and attached to NoSQL database.
Ex: CRUD ...

MICROSERVICES

Provides shopping cart operation functionality and generates orders by events.
Ex: add, remove and checkout ...

MICROSERVICES

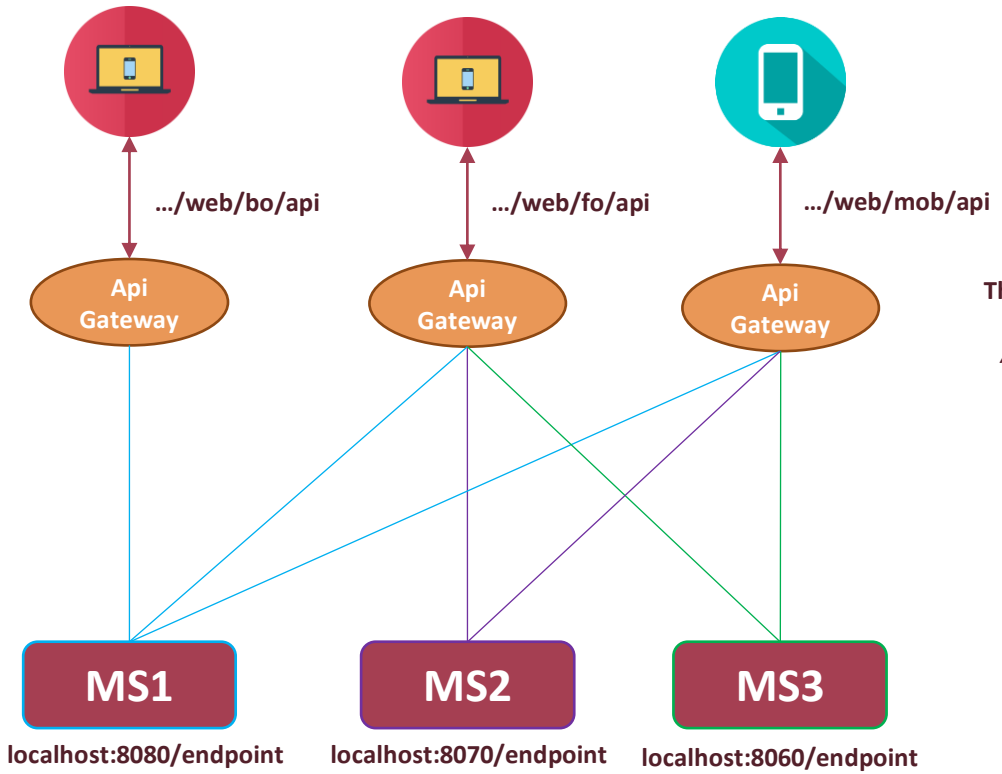
Receives requests from the cart service via event bus and attached NoSQL db.
Ex: Create, Updating Status...

MICROSERVICES

Provides notification operation functionality and attached to NoSQL database.
Ex: Send emails, Socket and firebase notifications...

BFF VS API GATEWAY

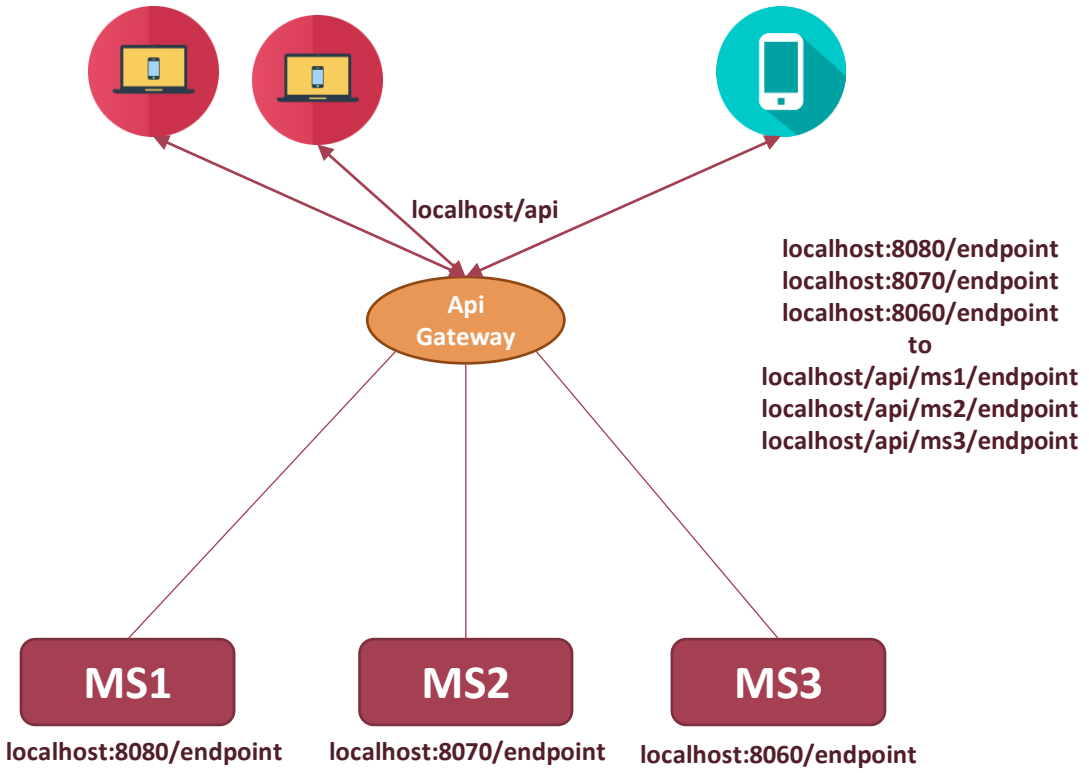
Multiple API Gateways provide separate APIs for each client



This is a service that provides a single-entry point

ApiG: Maintain a single api domain

API Gateway provides the same API for all clients

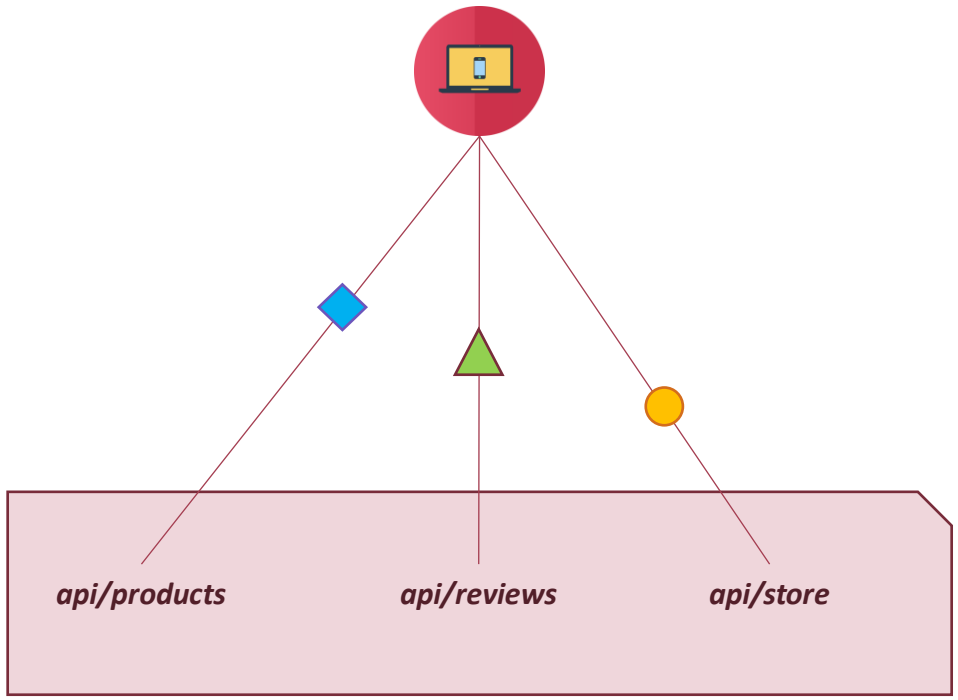


REST VS GRAPHQL

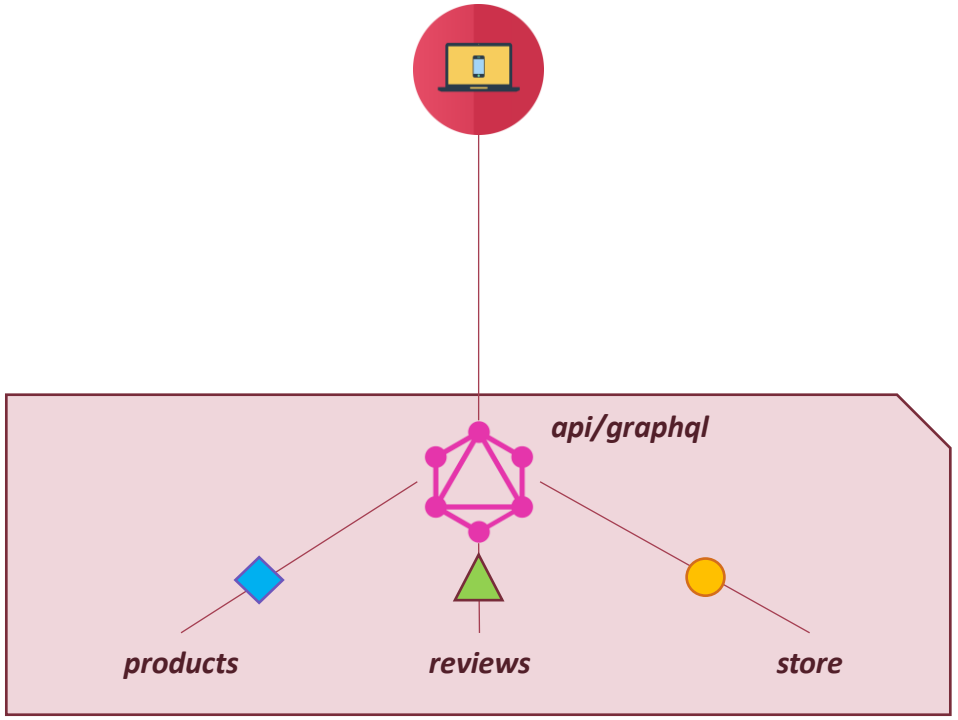
- ✓ One of the major differences is that with GraphQL, you only have one endpoint.
- ✓ With a single request. you can get an object and its related objects, while in REST you have different endpoints that can access different resources which means that if you need data from different resources you have to make different calls.
- ✓ For Rest each call returning complete objects probably with data you don’t even need…

REST VS GRAPHQL

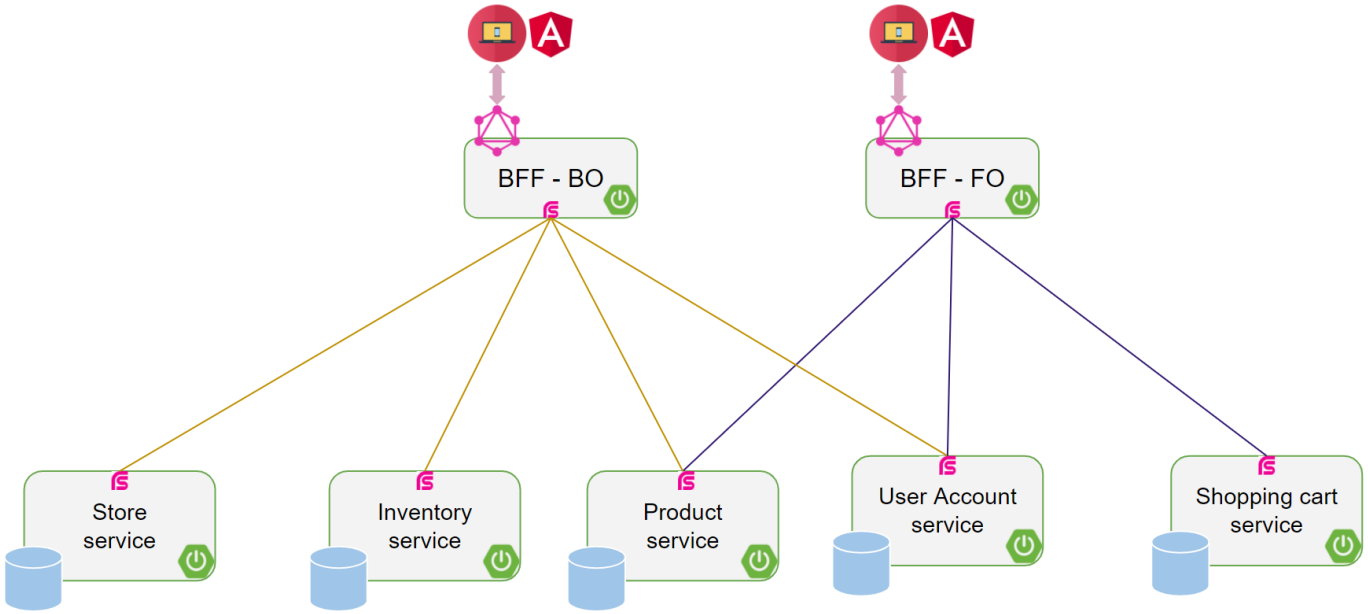
REST



GRAPHQL



COMMUNICATION

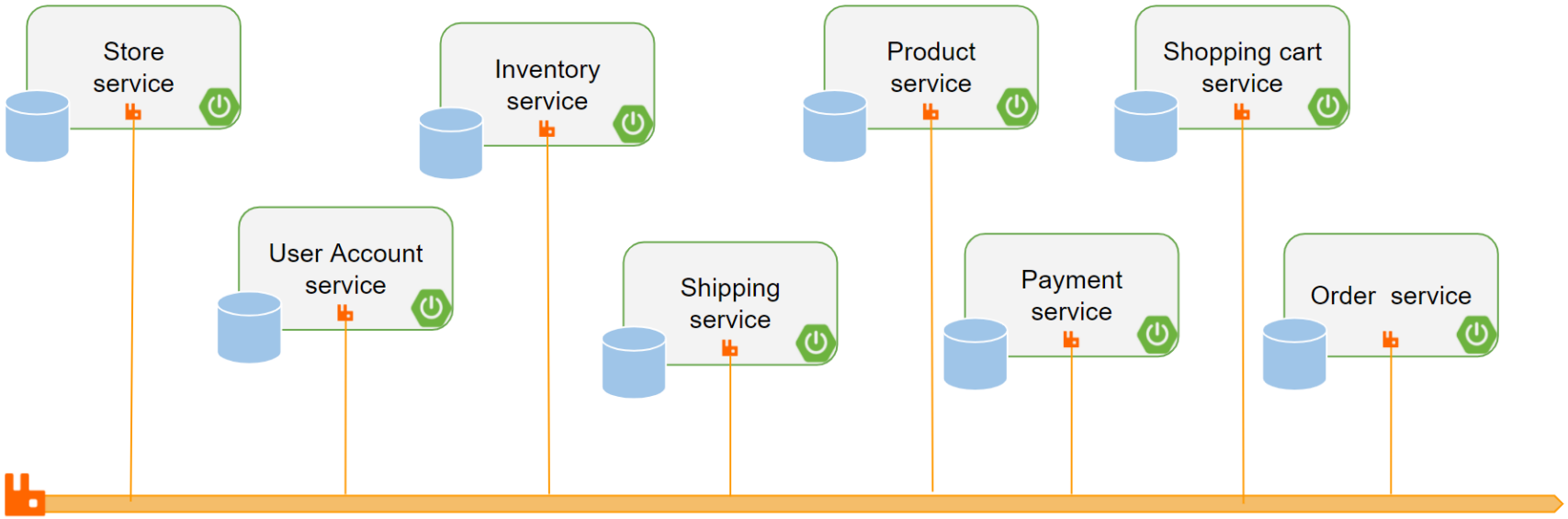


- RSocket is a new, message-driven, binary protocol that standardizes the approach to communication in between microservices. support for multiplexing and binarized payloads

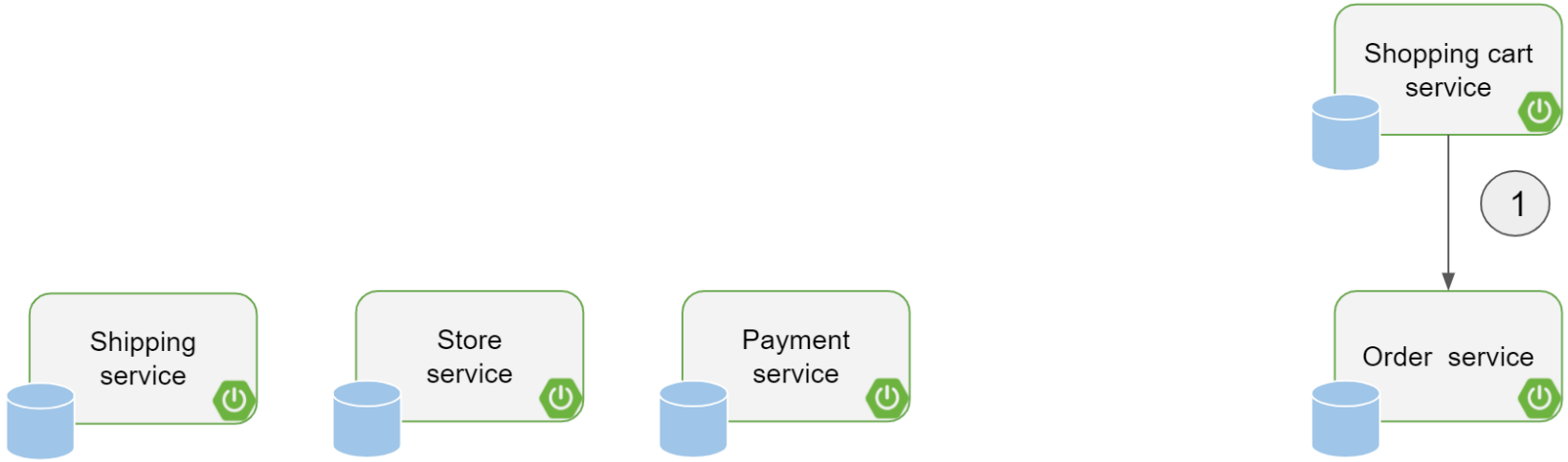
- Interaction model of HTTP problems: the server has to send a response back to the client, even if the client is not interested in processing it. The size of data is higher than in the case of binary protocols.

DATA CONSISTENCY

- The communication is made by sending messages that contain information or commands that need to be processed. The sender is called the Producer.
- These messages are stored (in memory or persisted) in a queue and processed by another microservice (called the Consumer). Then, once a message is processed, it is removed or dequeued, which assures that it is processed only once.

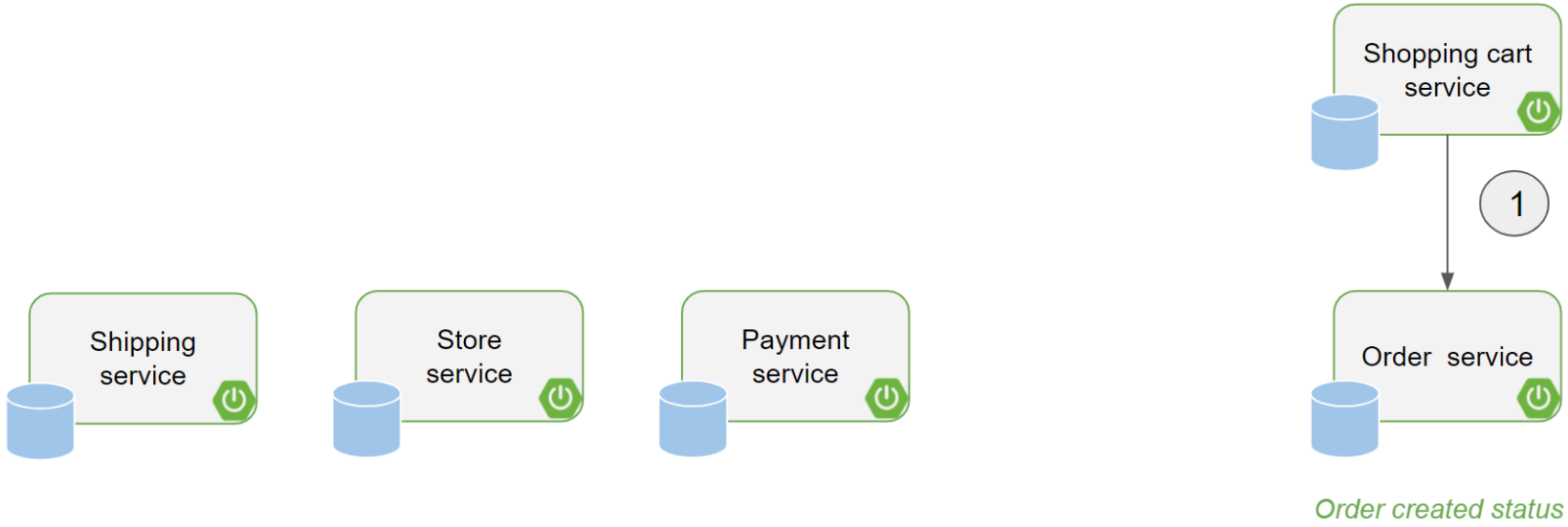


SAGA PATTERN



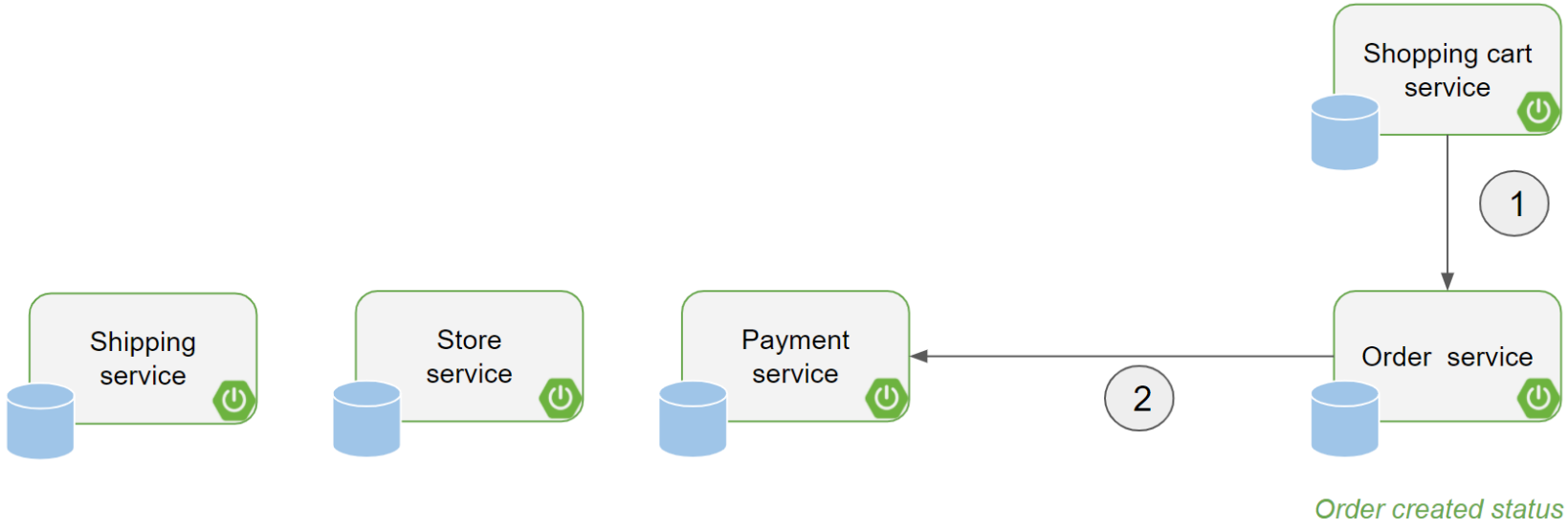
1. The order services triggered first after checkout

SAGA PATTERN



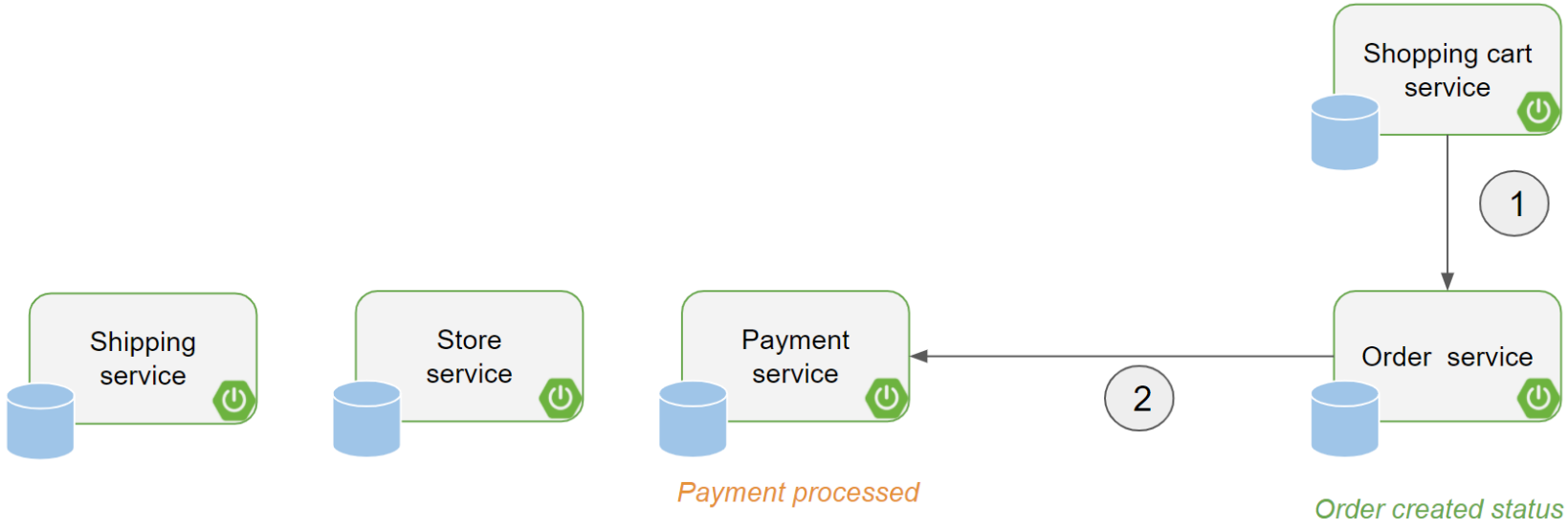
1. The order services triggered first after checkout

SAGA PATTERN



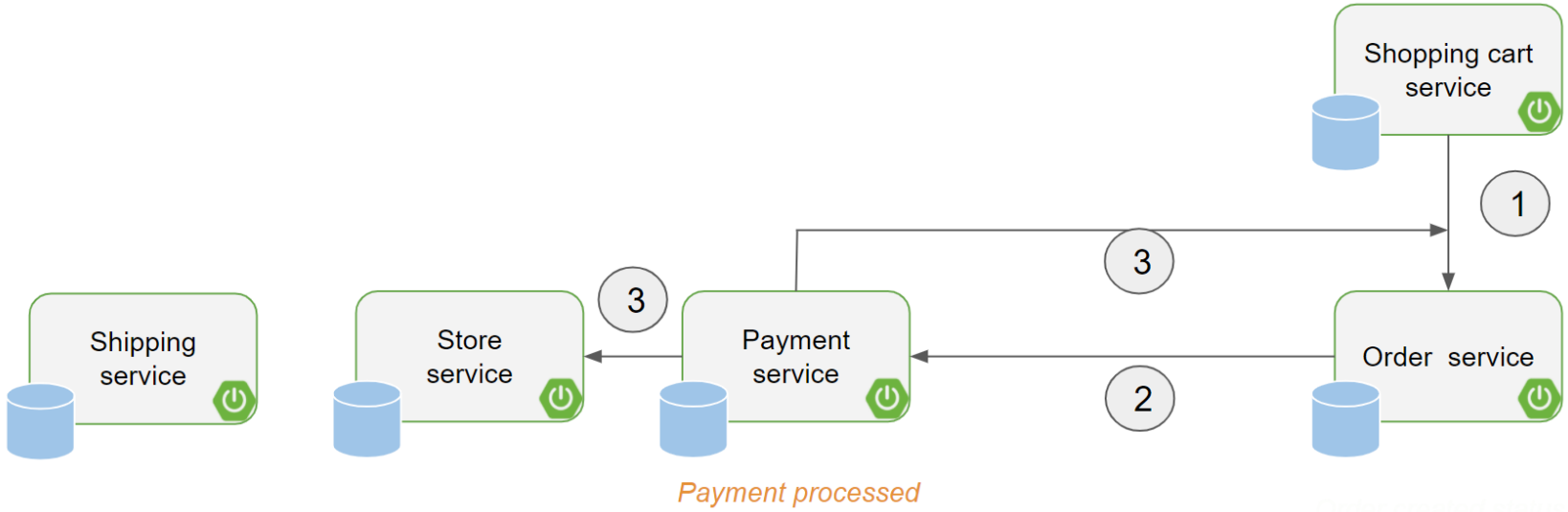
1. The order service triggered first after checkout
2. The order service trigger the payment service and payment service does the payment for this particular order

SAGA PATTERN



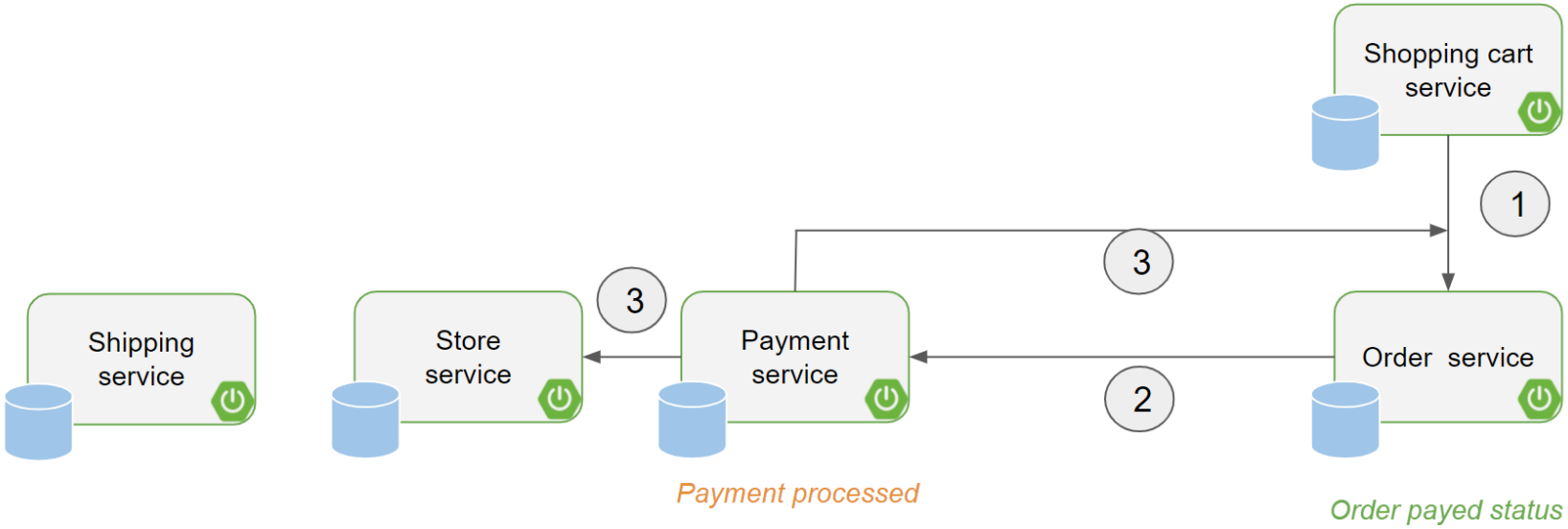
1. The order service triggered first after checkout
2. The order service trigger the payment service and payment service does the payment for this particular order

SAGA PATTERN



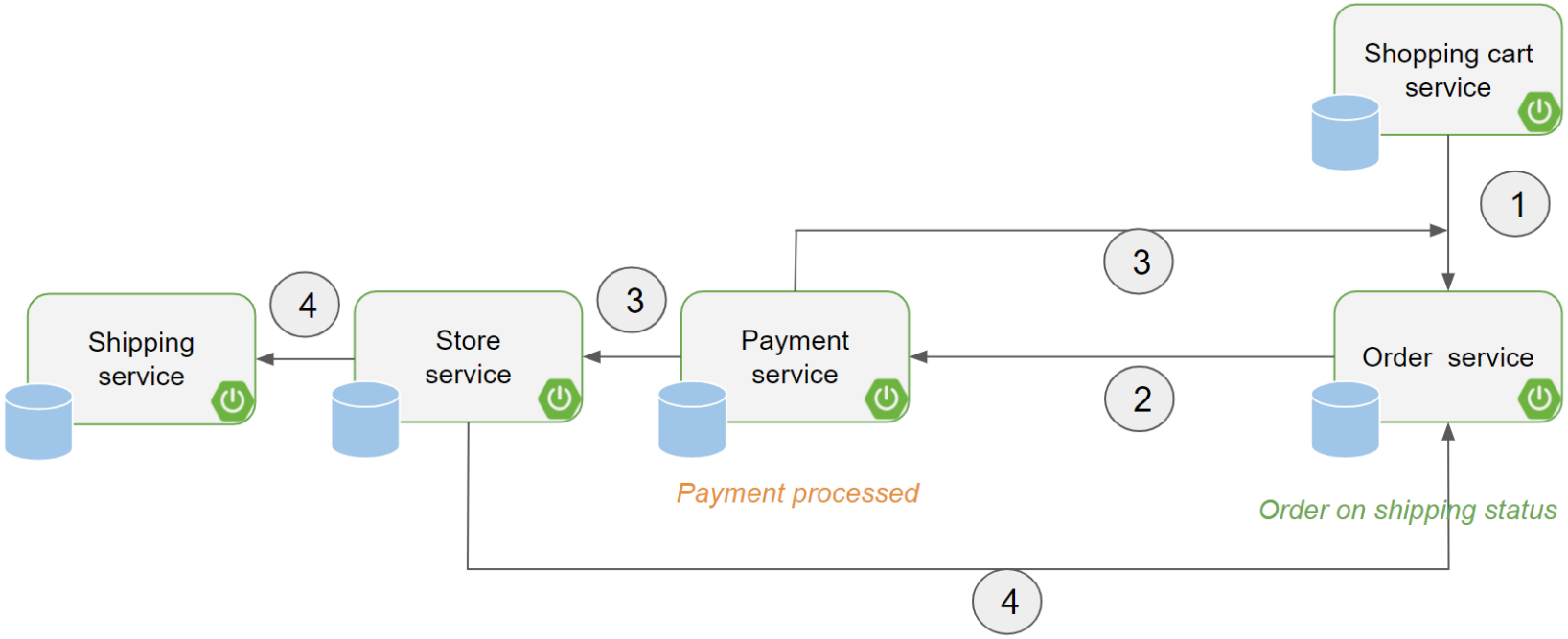
1. The order service triggered first after checkout
2. The order service trigger the payment service and payment service does the payment for this particular order.
3. Once the payment finishes the transaction,it notifies the order service and the store

SAGA PATTERN



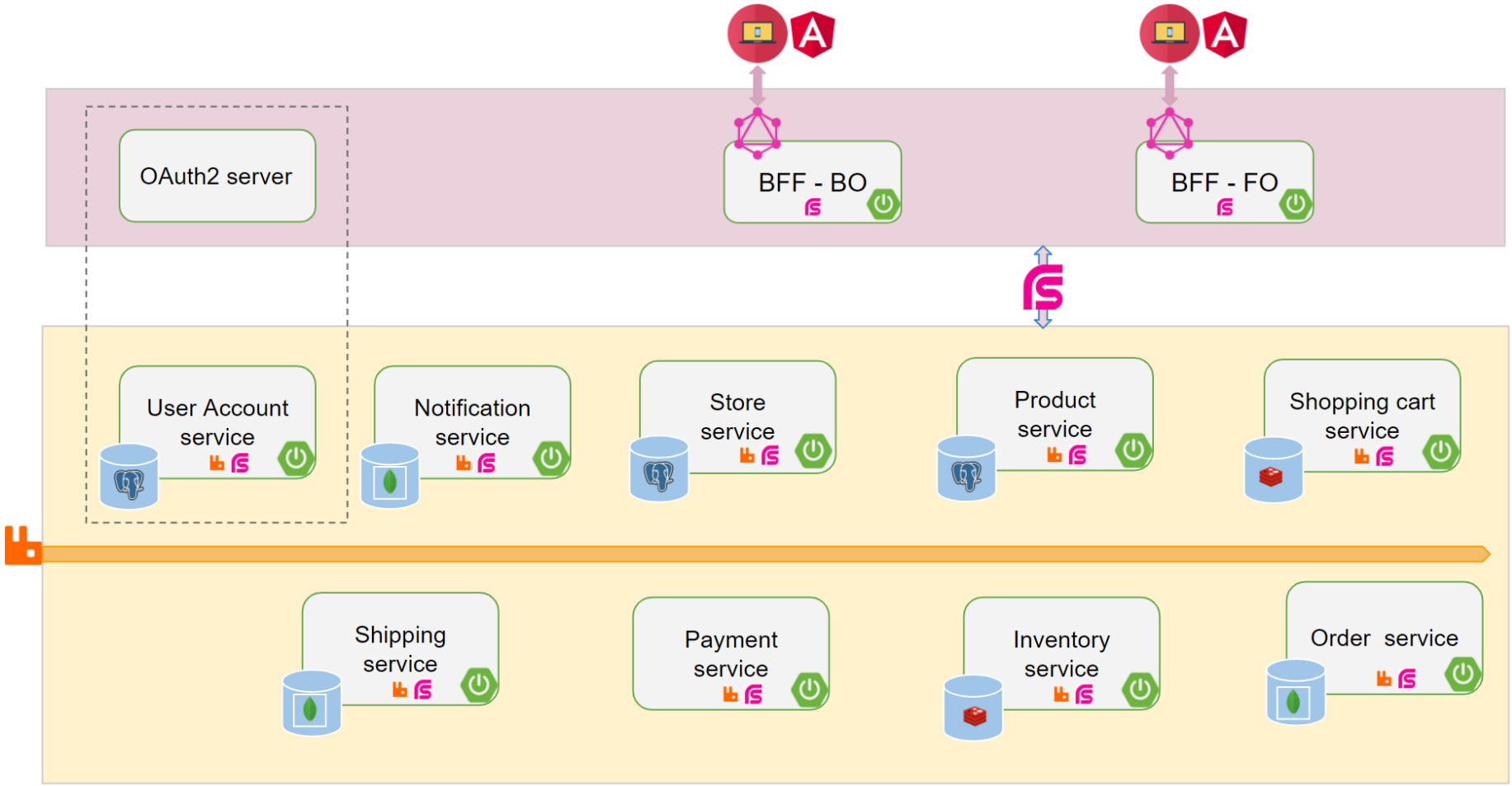
1. The order service triggered first after checkout
2. The order service trigger the payment service and payment service does the payment for this particular order.
3. Once the payment finishes the transaction,it notifies the order service and the store

SAGA PATTERN



1. The order service triggered first after checkout
2. The order service trigger the payment service and payment service does the payment for this particular order.
3. Once the payment received the transaction, it notifies the order service and the store.
4. After the store service finish booking this order, it trigger the shipping service and order service

SHOPPING PORTAL



System.out.print(“THE END”);

THANKS