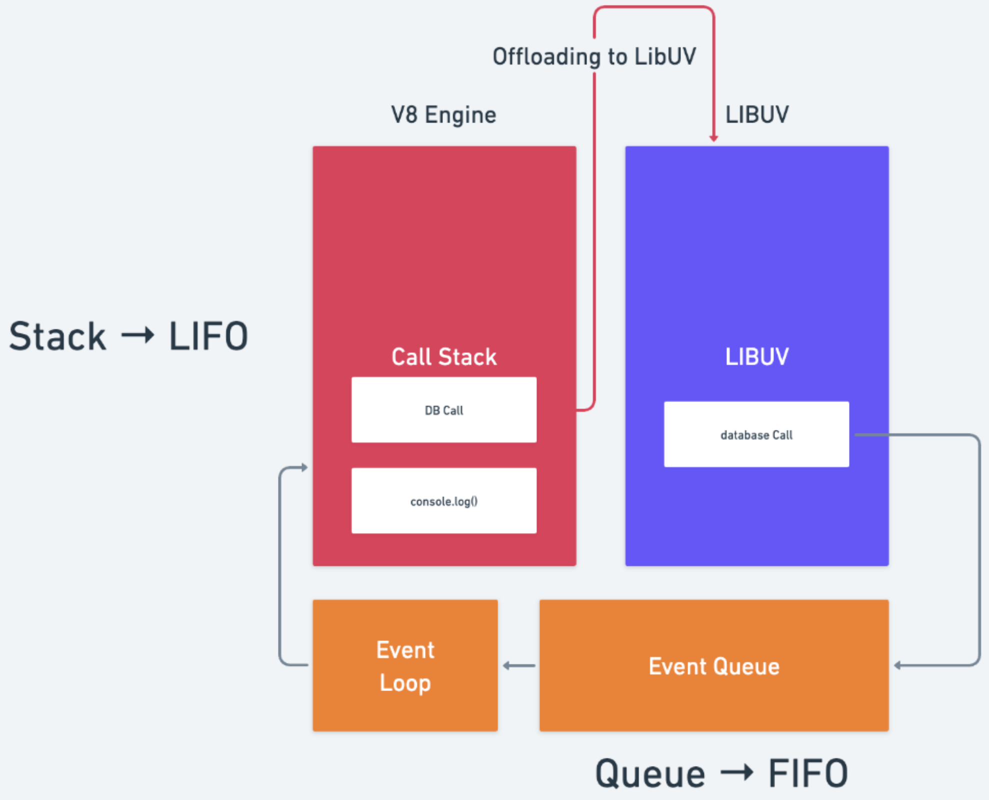
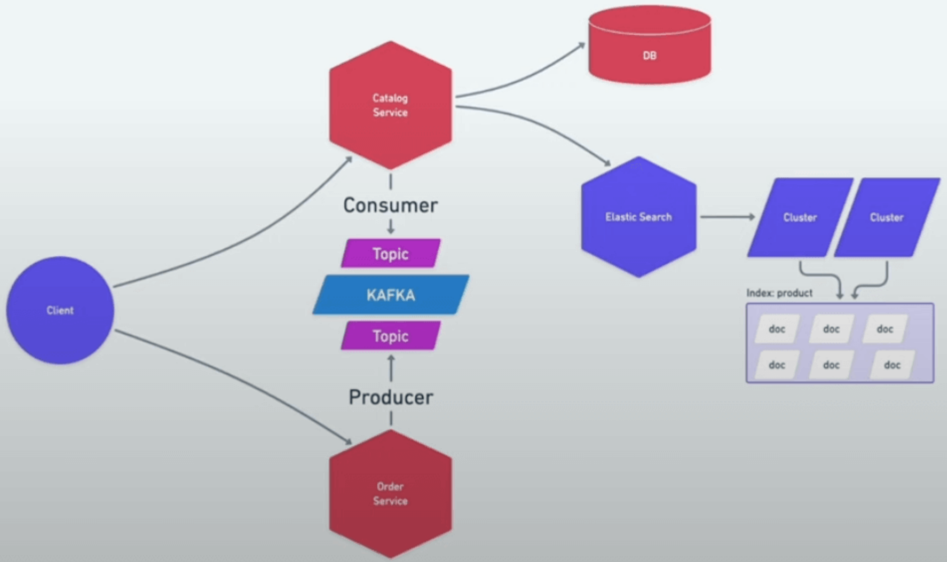
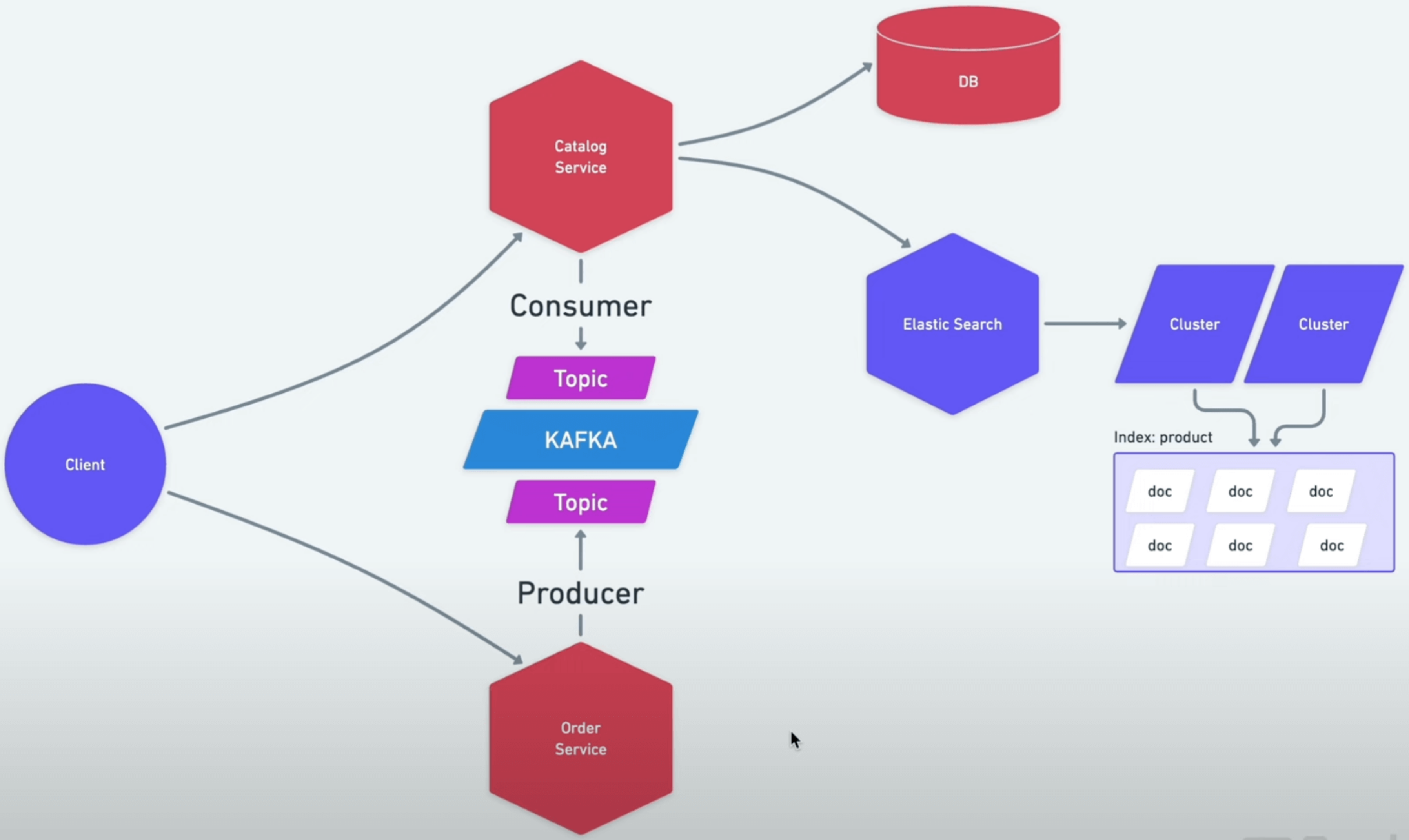
Node Ecosystem

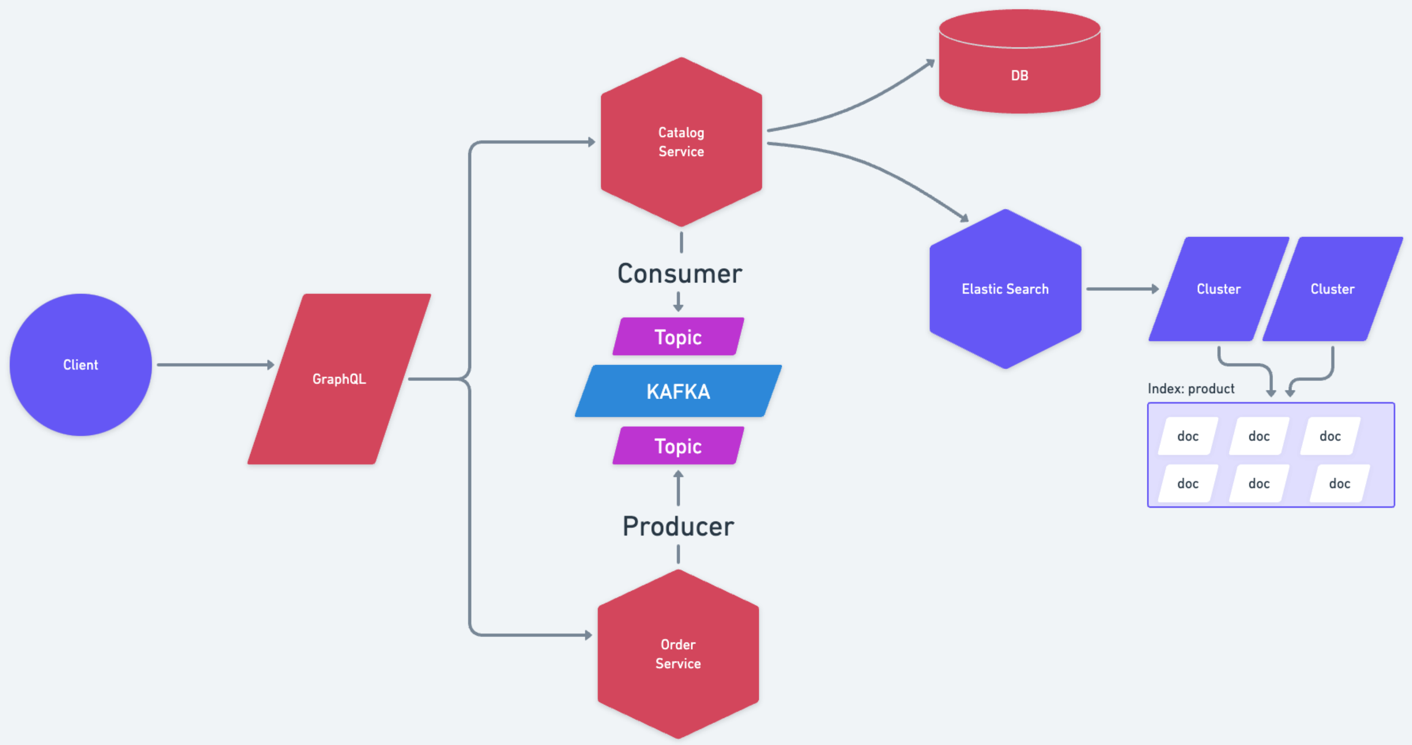


Node.JS Microservice Using

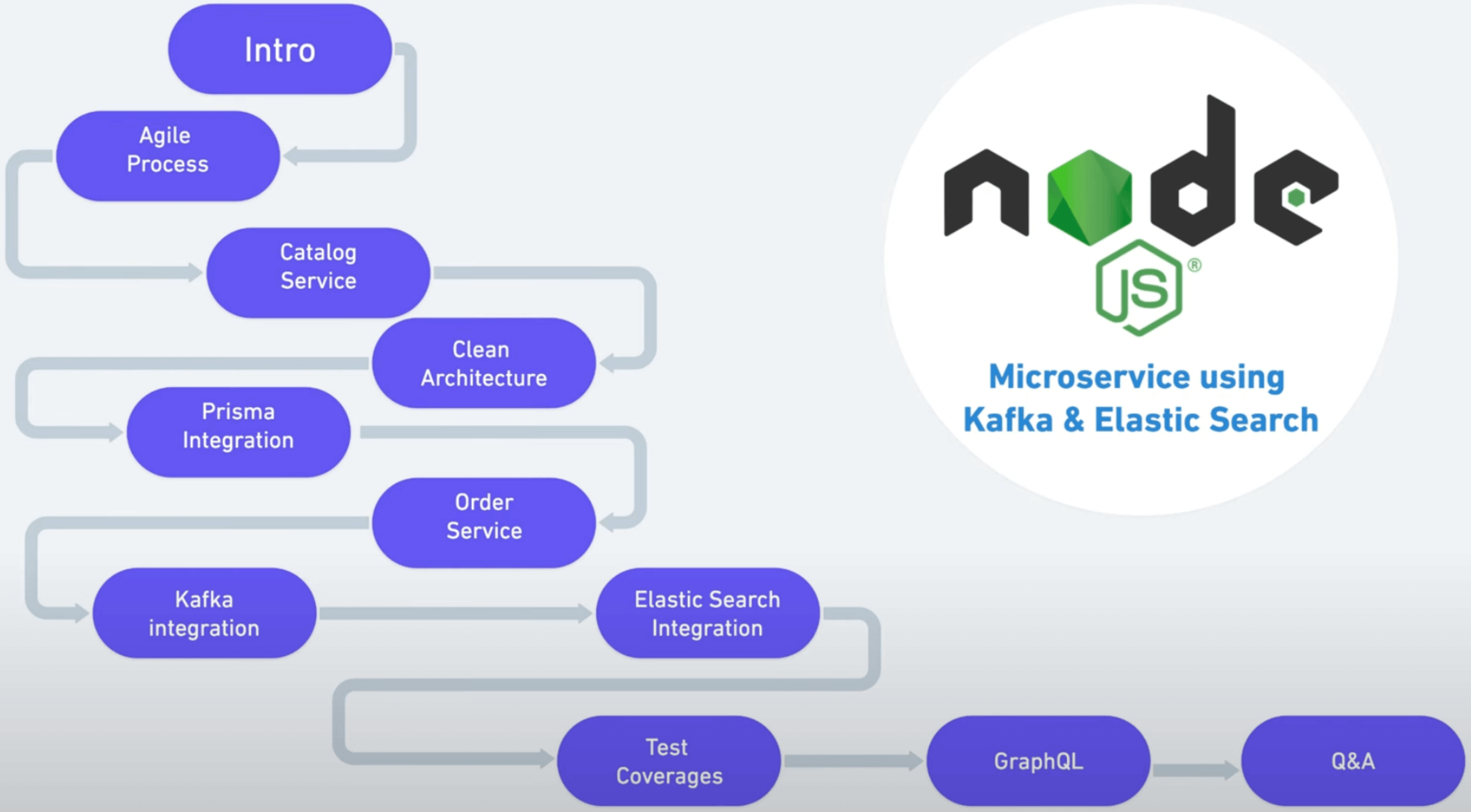
- Kafka
- Elastic Search
- GraphQL




Catalog Service
DB
Elastic Search
Cluster
Cluster
Index: product
doc
doc
doc
doc
doc
doc
Consumer
Topic
KAFKA
Topic
Producer
Client
Order Service

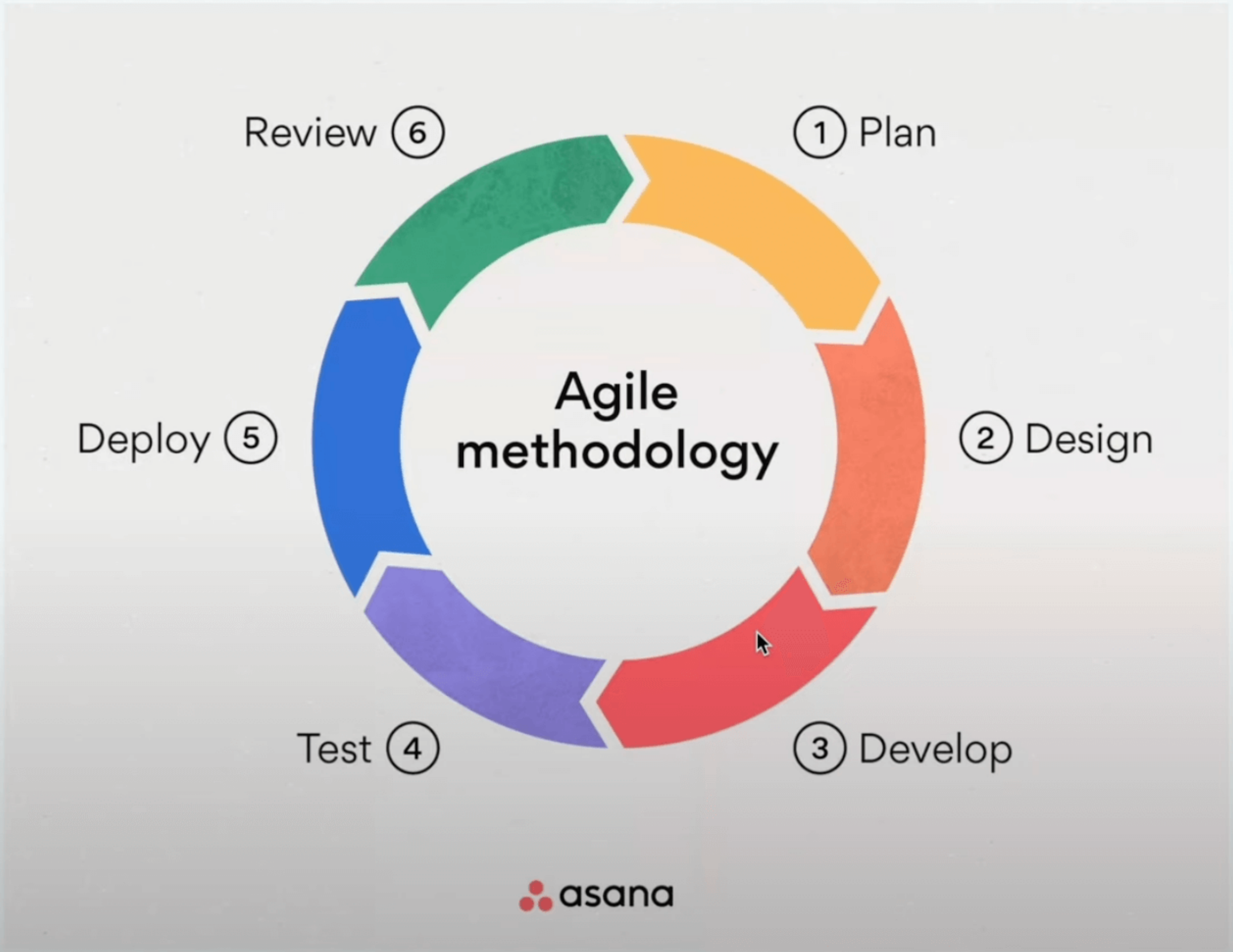


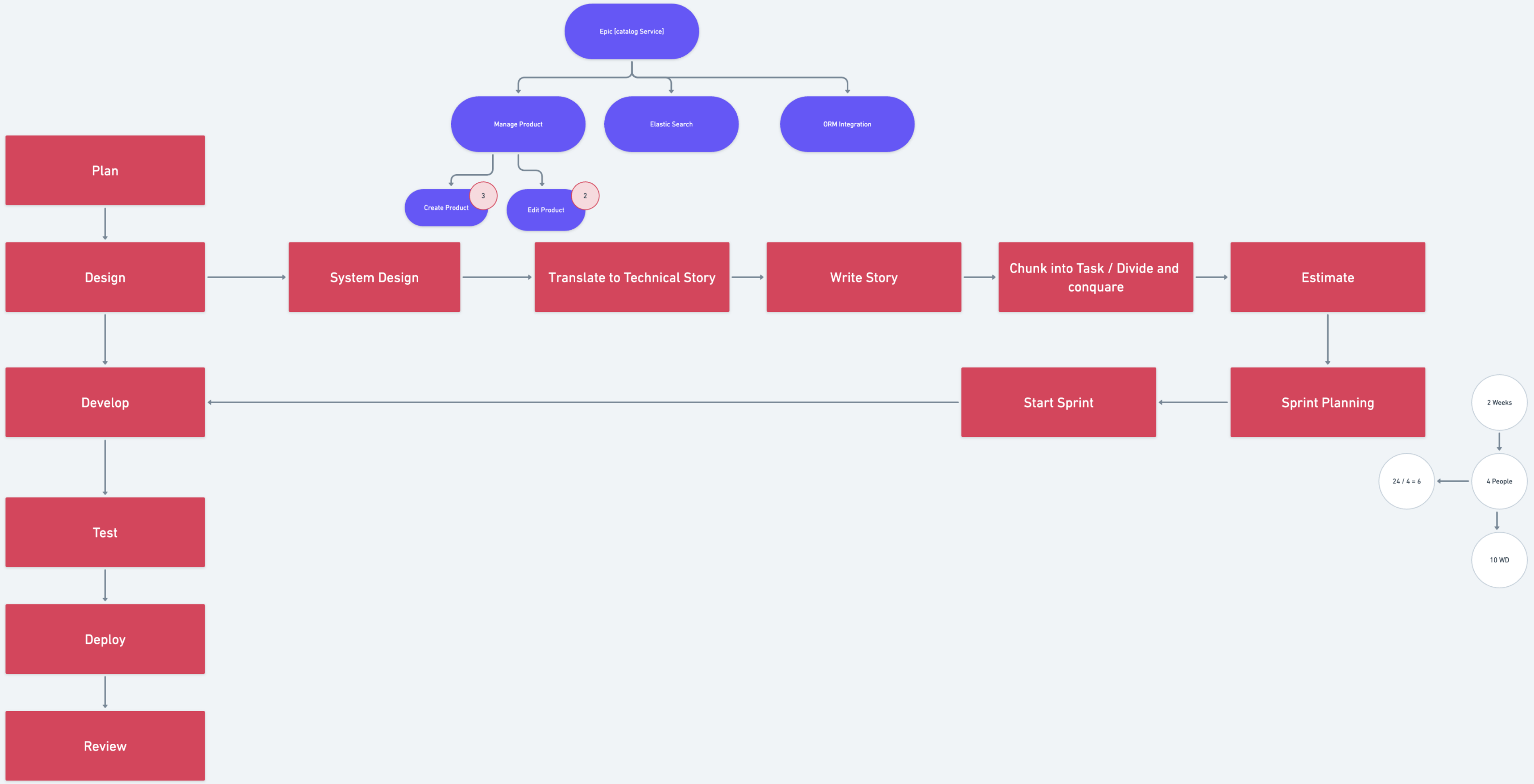
- Create product
- Update / delete
- Elastic search Integration
- Update Stock



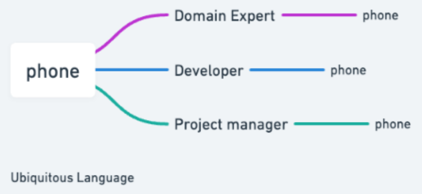
Microservice using
Kafka & Elastic Search

Agile Process




Epic [catalog Service]
Manage Product
Elastic Search
ORM Integration
Create Product
3
Edit Product
2
Plan
Design
System Design
Translate to Technical Story
Write Story
Chunk into Task / Divide and conquare
Estimate
Develop
Start Sprint
Sprint Planning
2 Weeks
24 / 4 = 6
4 People
10 WD
Test
Deploy
Review

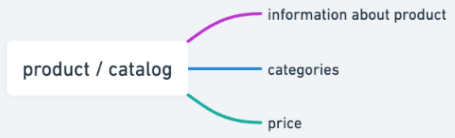
Identify System Boundaries: DDD

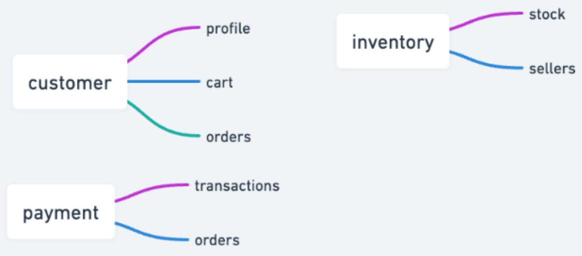


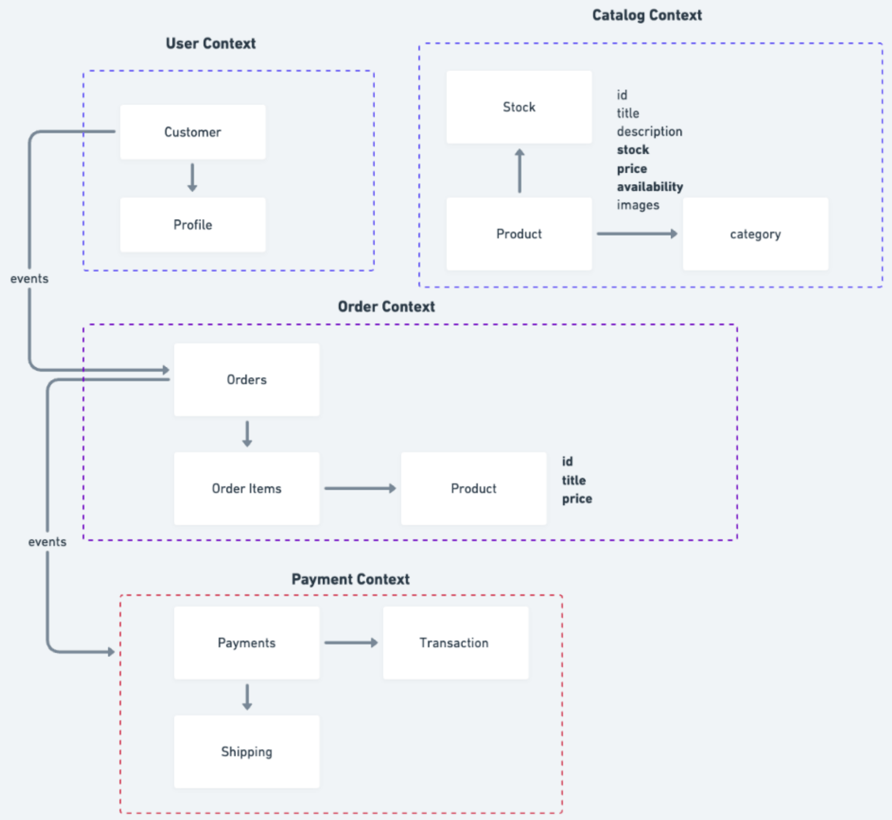
Domain
E-Commerce Application

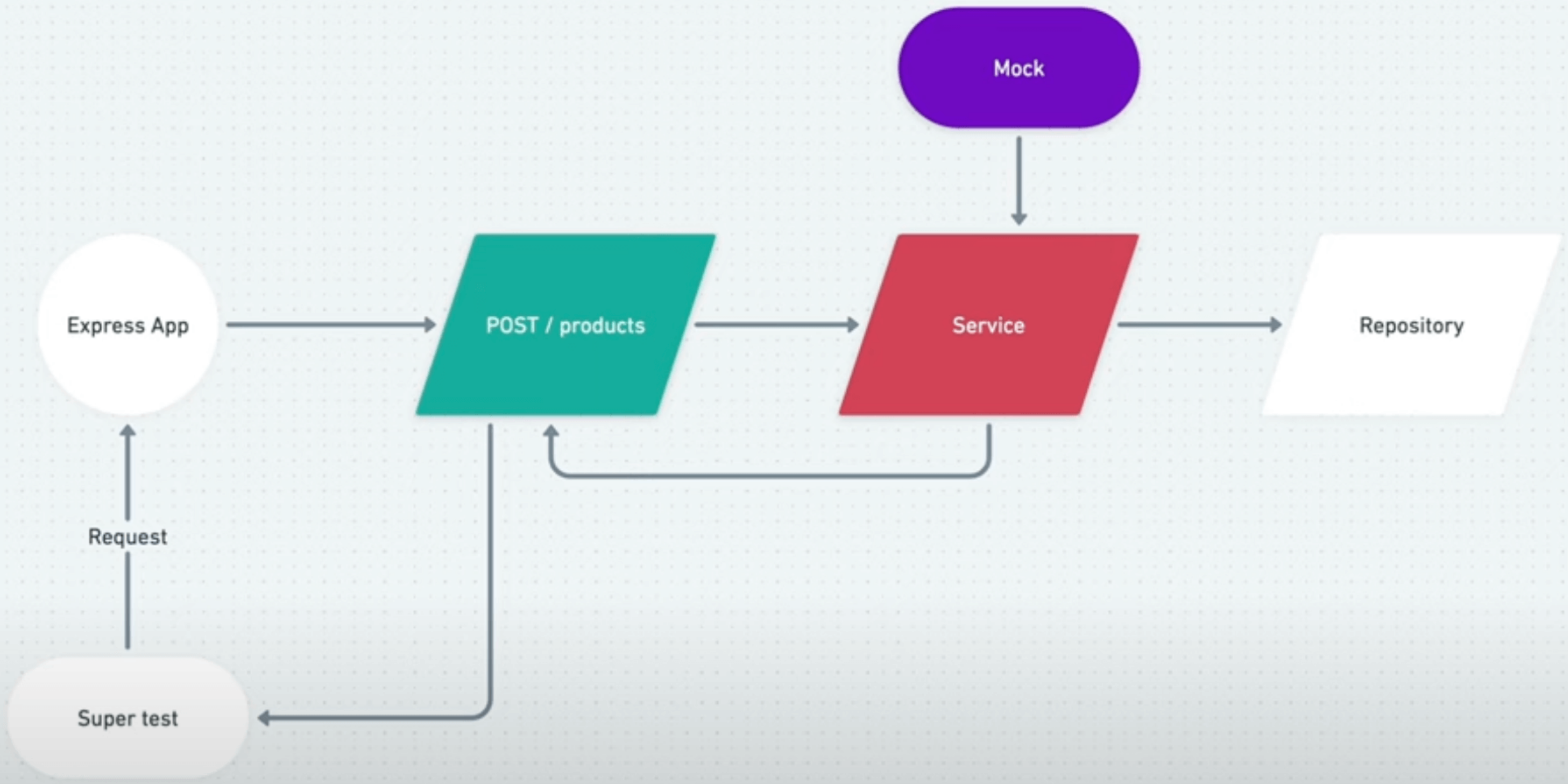
Let's Figure out the Logical Boundaries so we can have our microservices their own isolated Boundary and responsibility to perform operation autonomously. i.e: Each microservice can responsible to it's own Domain and Models if any dependency or interaction need it will be through Message broker.

Domain
E-commerce
?

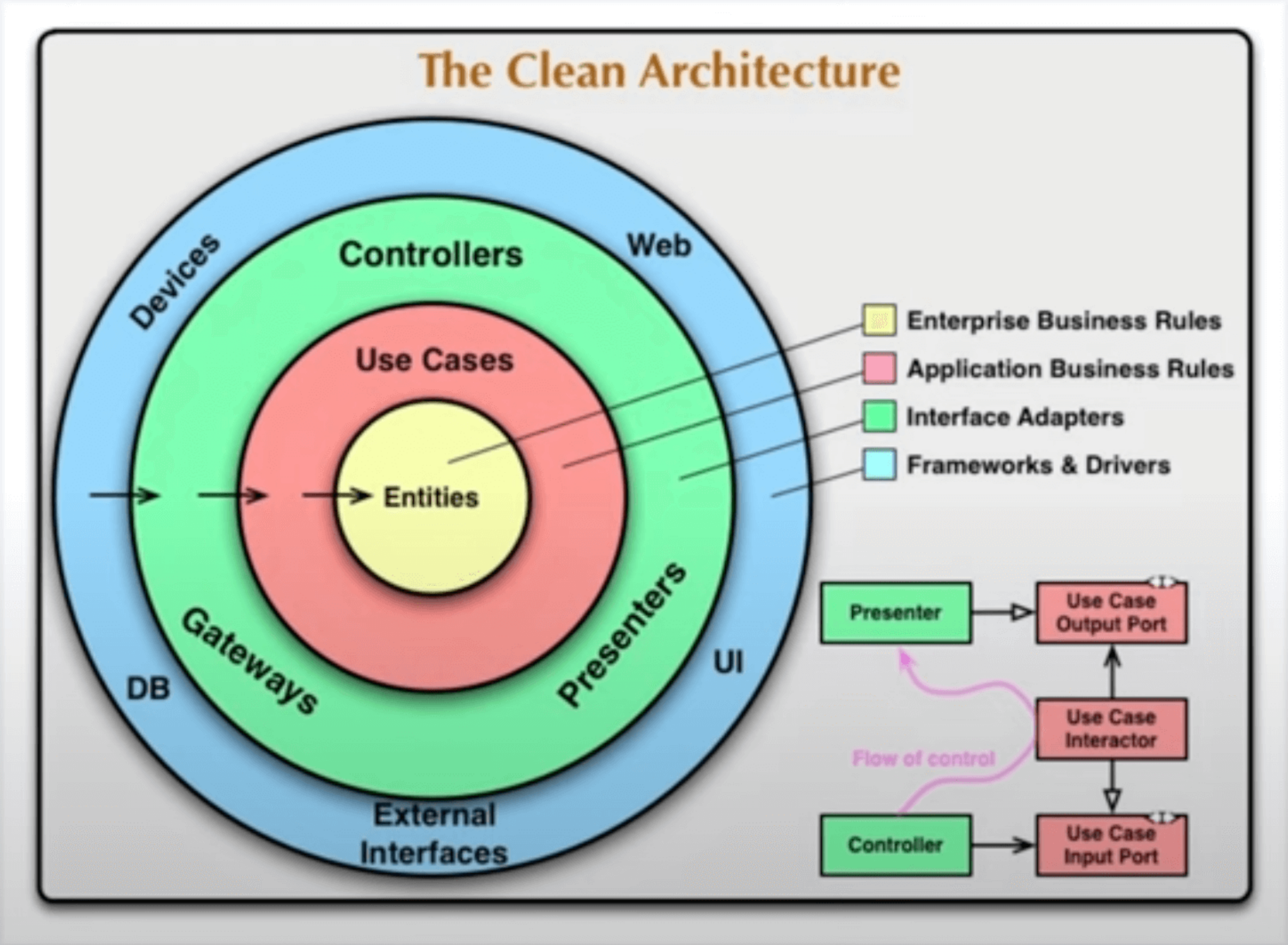


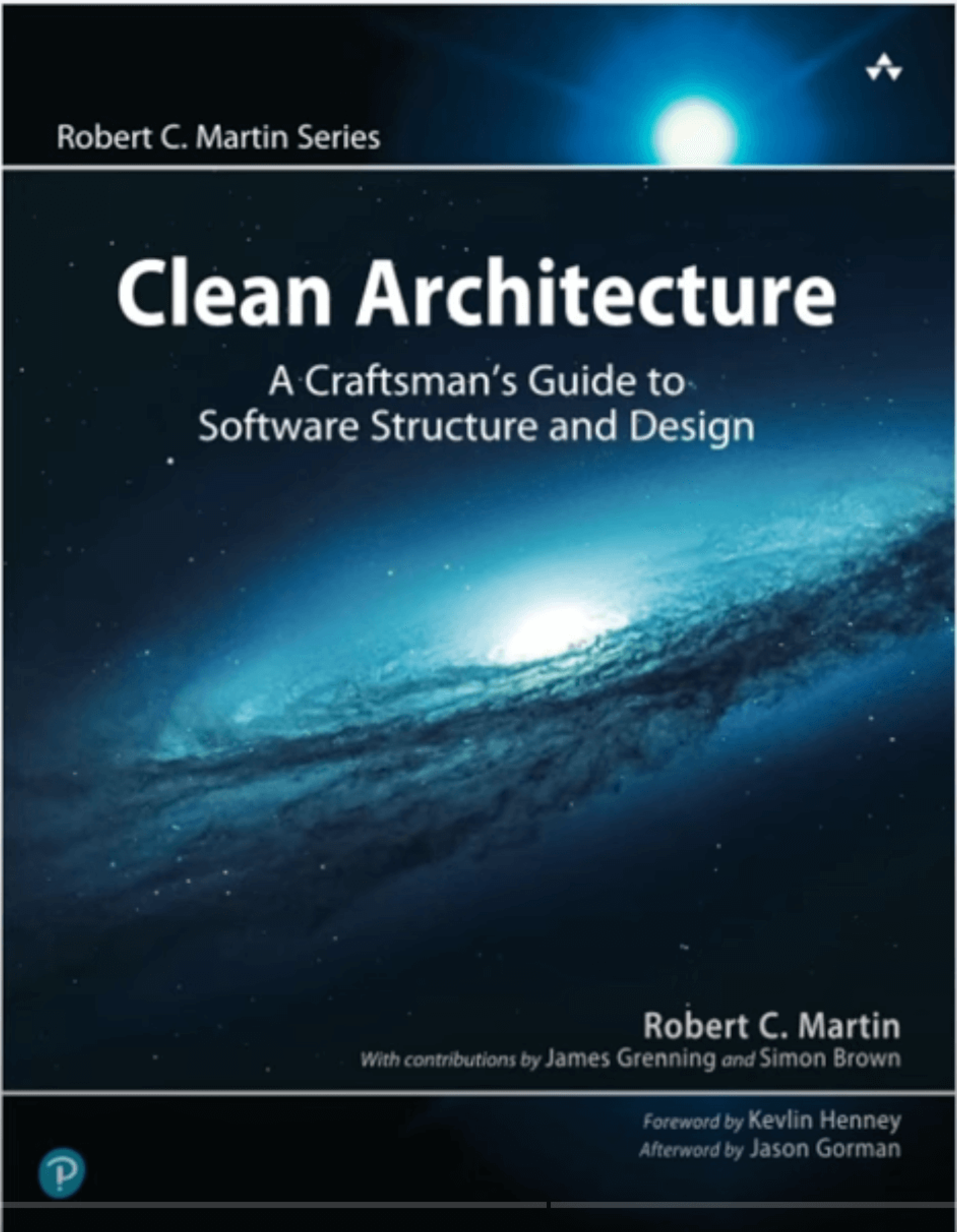





Mock
Express App
POST / products
Service
Repository
Request
Super test

Clean Architecture





Robert C. Martin Series

Clean Architecture

A Craftsman's Guide to Software Structure and Design

Robert C. Martin

With contributions by James Grenning *and* Simon Brown

Foreword by Kevlin Henney

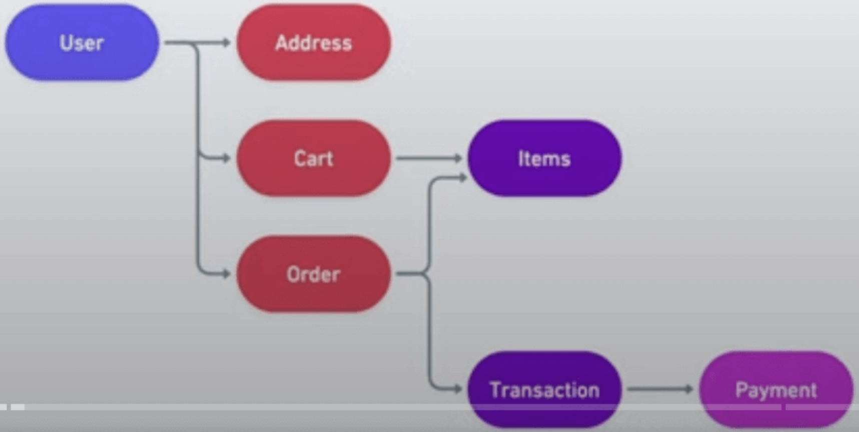
Afterword by Jason Gorman

1. **Independent of Frameworks.** The core business logic and rules should not be dependent on the frameworks or external tools. This allows for flexibility in choosing and changing the tools without affecting the core business logic.
2. **Testable.** The business rules can be tested without the UI, Database, Web Server, or any other external element.
3. **Independent of UI.** The UI can change easily, without changing the rest of the system. A Web UI could be replaced with a console UI, for example, without changing the business rules.
4. **Independent of Database**. You can swap out Oracle or SQL Server, for Mongo, BigTable, CouchDB, or something else. Your business rules are not bound to the database.
5. **Independent of any external packages or entity**. In fact your business rules simply don't know anything at all about the outside world.

ORM

Why Need to use?

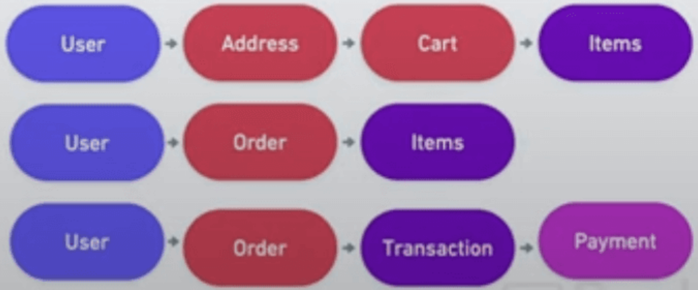
- When Not to put effort on writing SQL queries on Low-level Database Complexity
- When it comes to productive rapid development / Startup projects
- At any time, the Database can be changed based on Business requirements
- Not caring about manual sanitization to protect data. ORM will handle most of them.
- Get advantages of Type-safety Object Oriented programming
- Automatic Query generation and Object relation Mapping

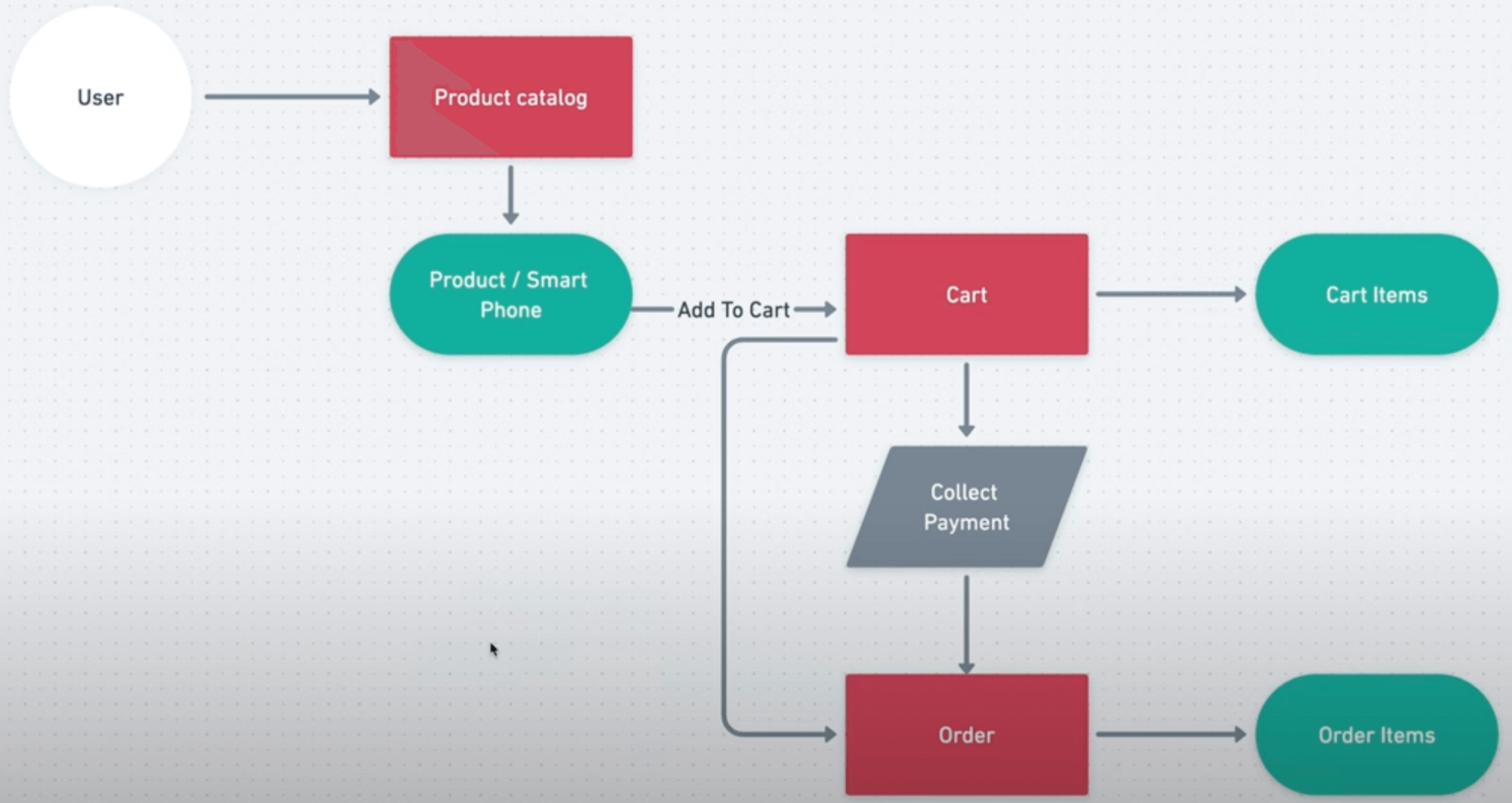


Native SQL

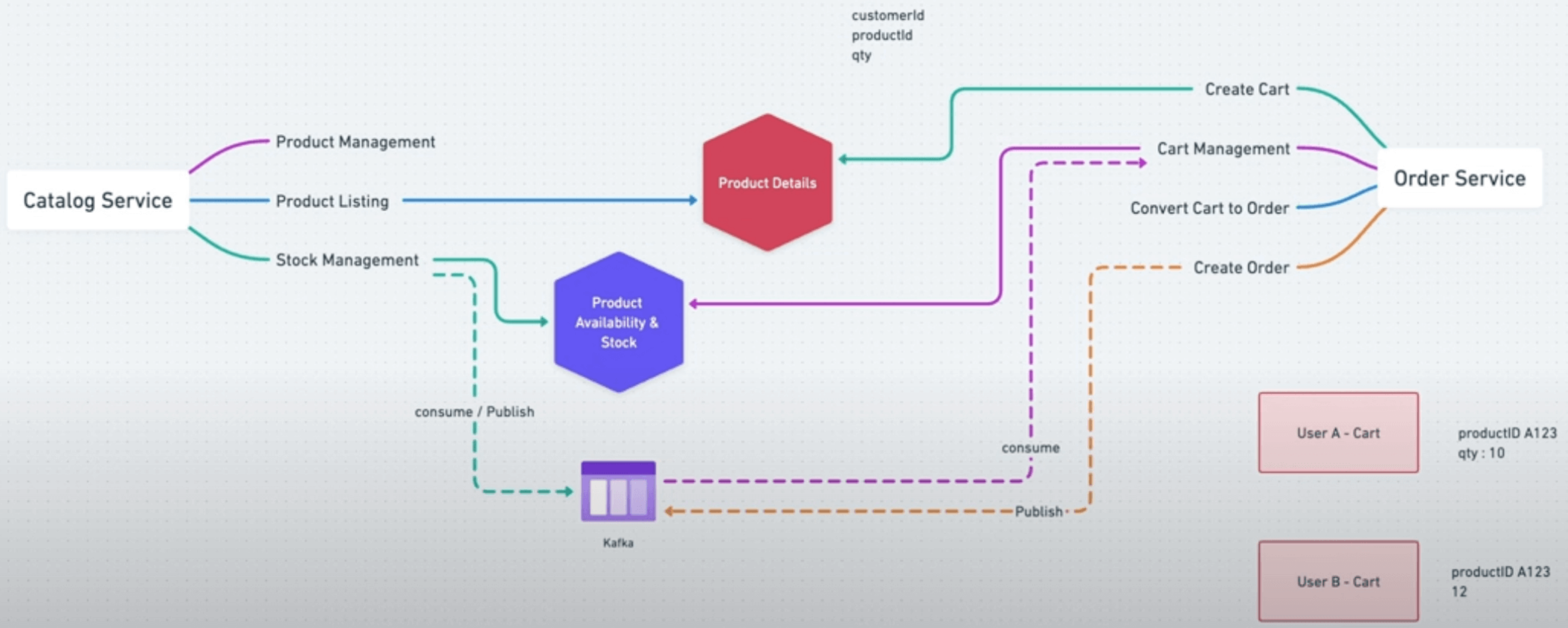
Why Need to use?

- When comes to performance always matter, ORM create overheads
- You need more control on SQL and flexibility
- When you have small application and simple feature only a couple of SQL Operations
- Maintaining and debugging is easy
- More focus on Data base operations to perform complex jobs, transactions with low level controls
- It Enhancing SQL and database knowledge

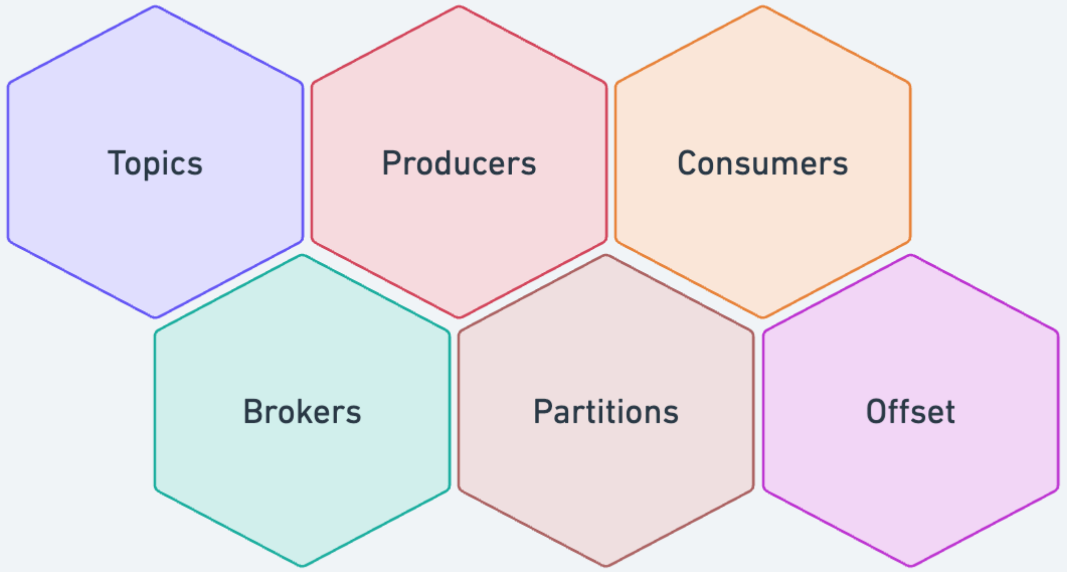



User
Product catalog
Product / Smart Phone
Add To Cart
Cart
Cart Items
Collect Payment
Order
Order Items

Order and Catalog Use Cases



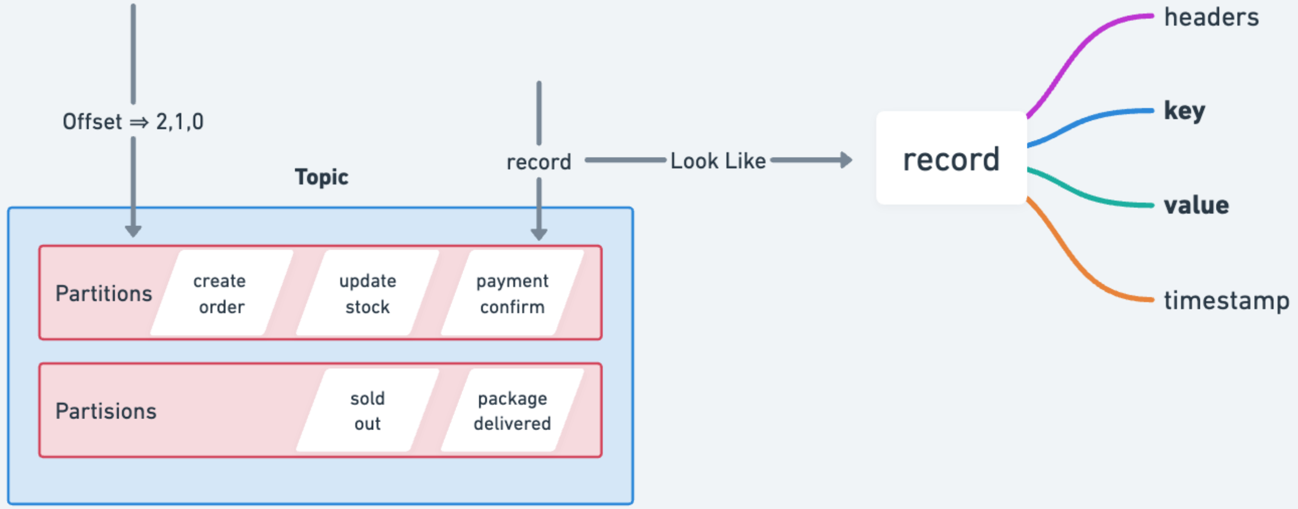
Core Components of Kafka



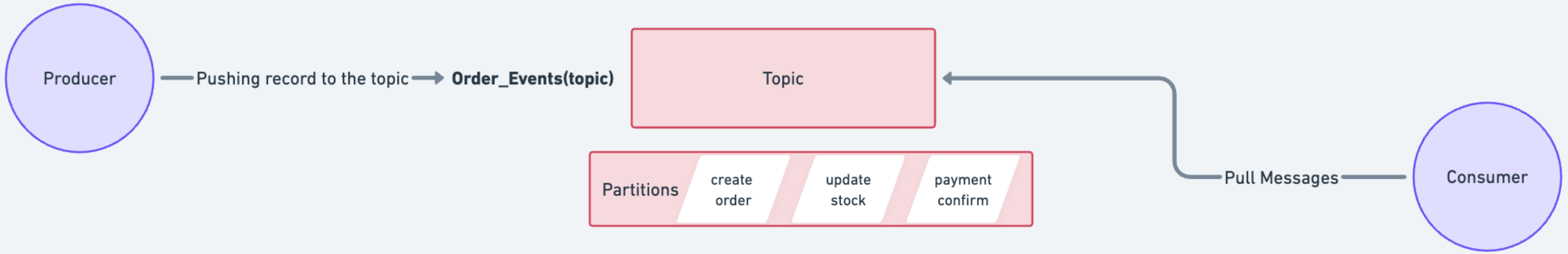
What is a Topic ?

A Topic is streams of Related messages in Kafka it is a Logical representation and group of categorised messages

Order_Events

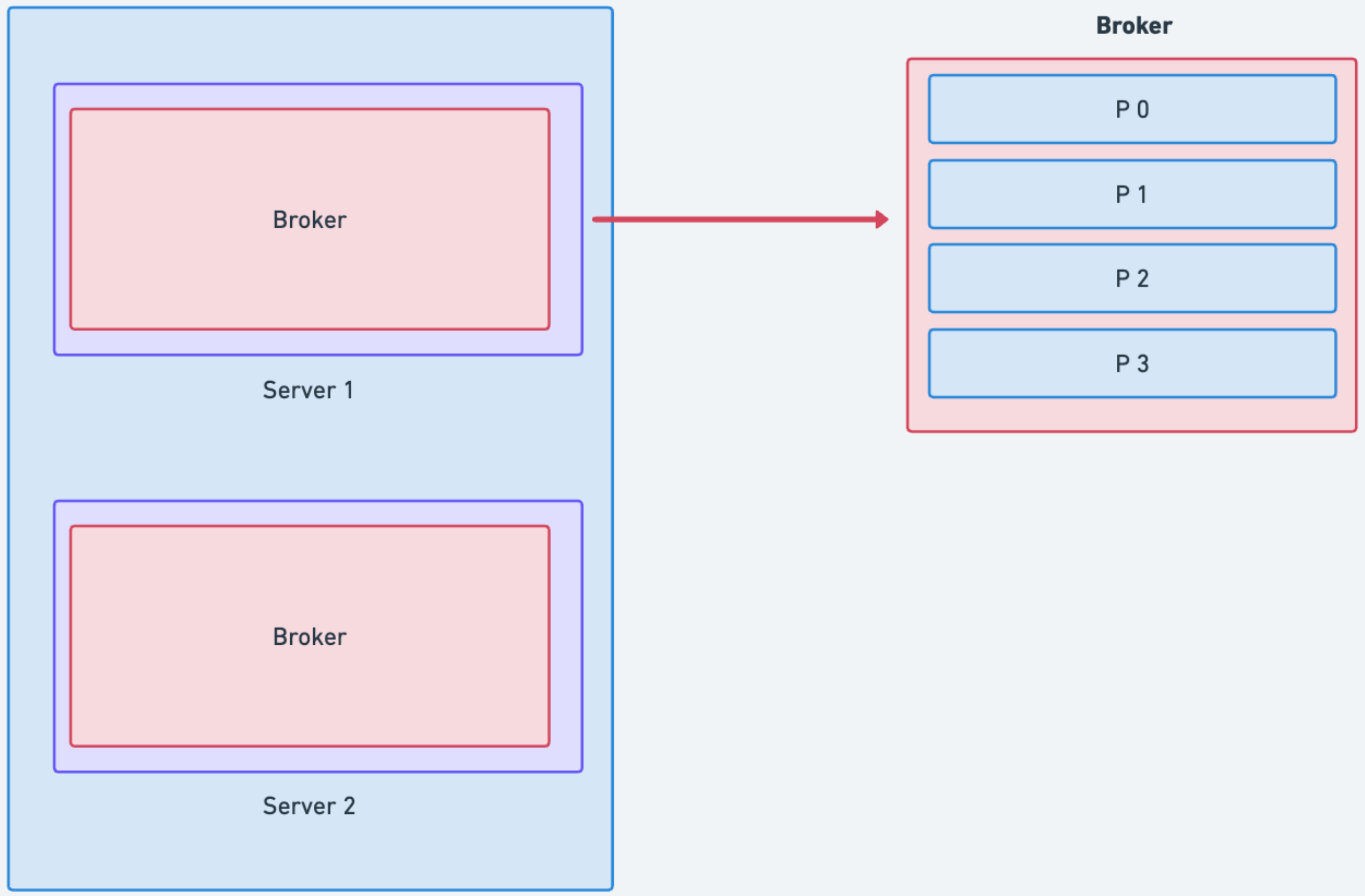


How Topic works ?

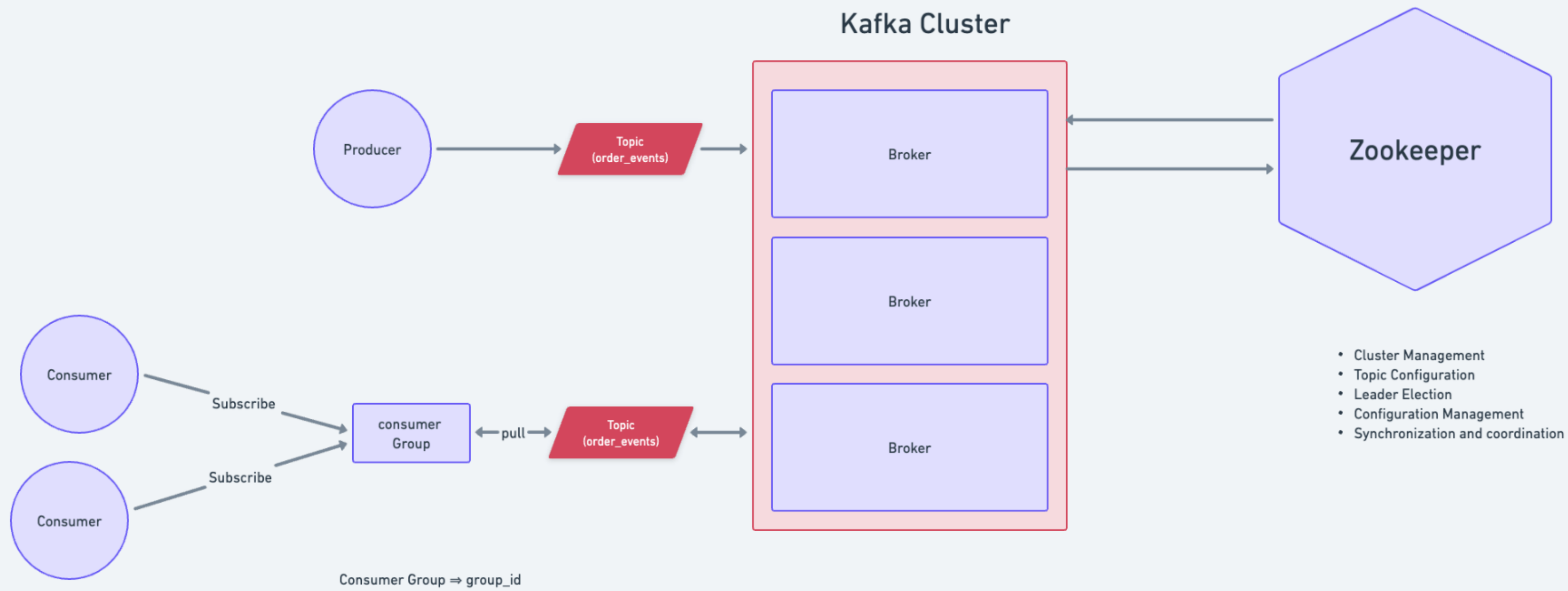


Where does it go?

What is Broker?
How Kafka Cluster is looks like?

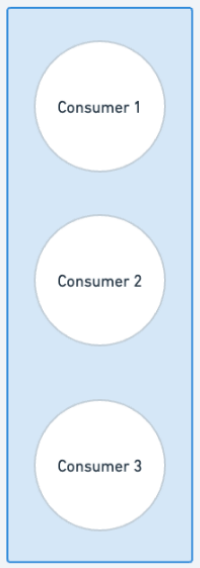


Let's have a look at Setup



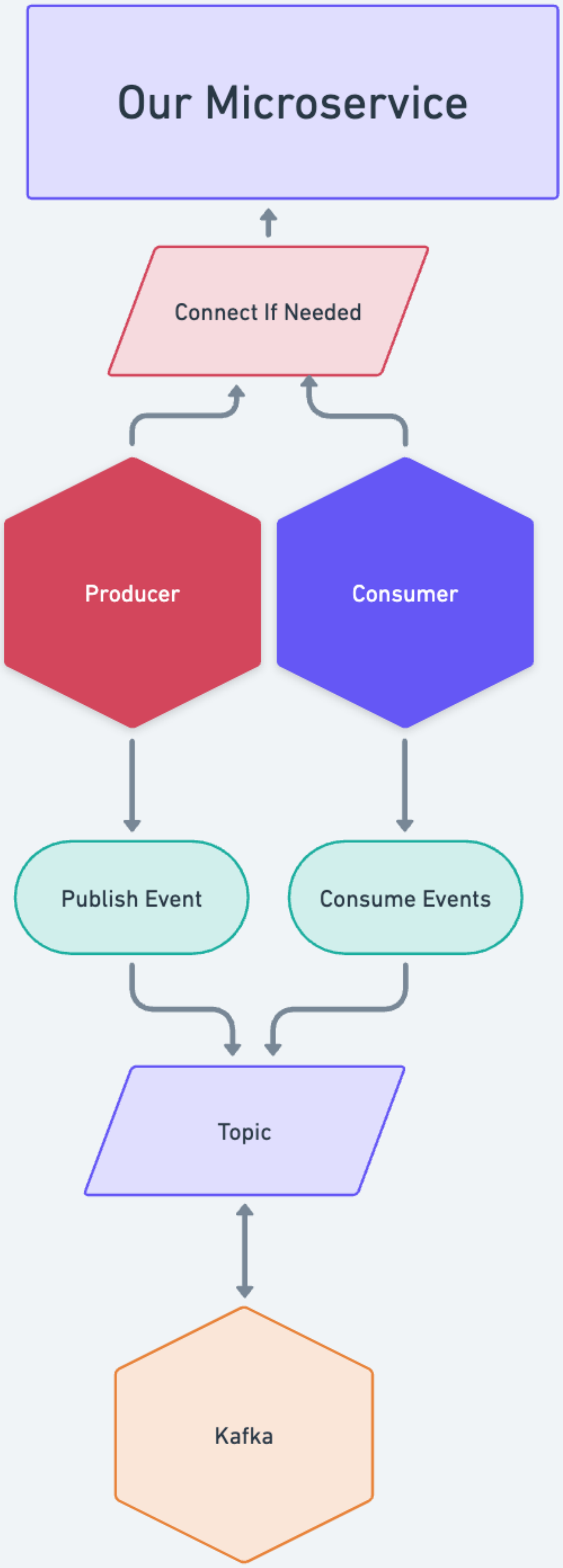
- Cluster Management
- Topic Configuration
- Leader Election
- Configuration Management
- Synchronization and coordination and handle Fault Tolerant mechanism

Consumer Group ⇒ group_id



Q&A

- How multiple instance of will handle same event ? example Create Order?
- What will Happen if consumers are down and producer sent and event ?
- When need to create topics and who is responsible to create it?


Our Microservice
Connect If Needed
Producer
Consumer
Publish Event
Consume Events
Topic
Kafka