Benefits of Using Microservice Architecture:

1. Single Responsibility
2. Autonomous(Dev independent , deployment independent)
3. Heterogenous technology stack.
4. Scale for most used microservice.
5. Resilience: if one goes down , no impact on whole system.
6. Small independent component
7. Cloud support , dynamically scaled up or down based on need

Challenges:

1. Network latency , fault tolerance, message serialization
2. Unreliable network, Handling asynchronous output, varying loads within application tiers, distributed transaction.
3. Configuration changes for all services with min effort.
4. Deploy multiple versions of same service, routing
5. How to disconnect MS , from the network when crashes.
6. Isolate failed MS
7. To avoid cascading failure
8. Service discovery in an elastic manner ,when services up & down
9. Aggregate logs.
10. Identify different steps of single request spread across multiple MS.

MS vs SOA:

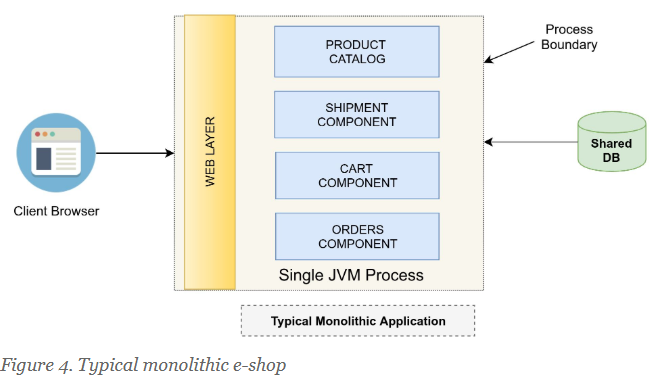
1. Both are distributed , high scalability
2. Both Use remote access protocol (REST,SOAP,JMS)
3. Both modular, high scalability.

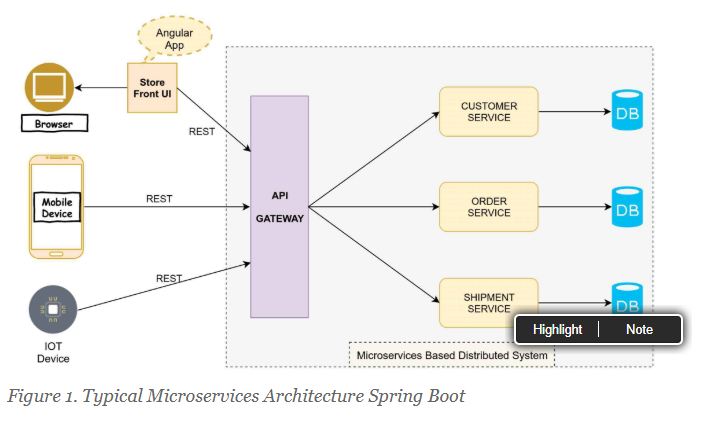
Difference:

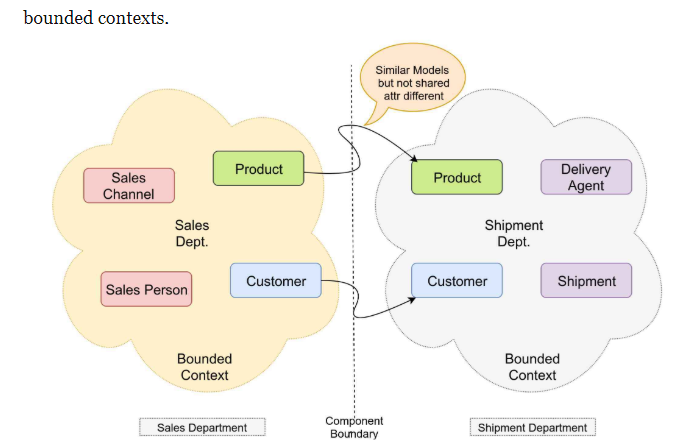
1. Communication: SOA:Enterprise service Bus MS : REST
2. Deployment : SOA : Monolithic MS : Automatic

Distributed Monolithic Anti pattern:

1. 2 diff MS , using synchronous call , causes total system failure.
2. Fix: use async calls



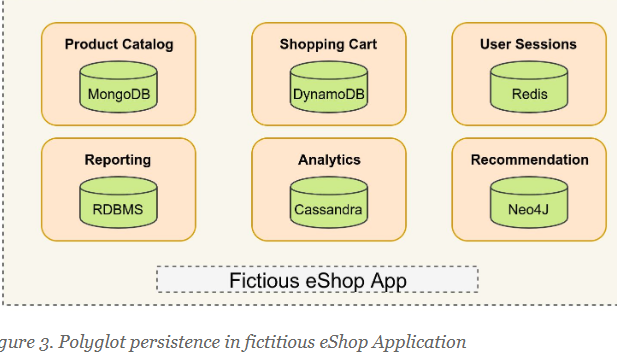




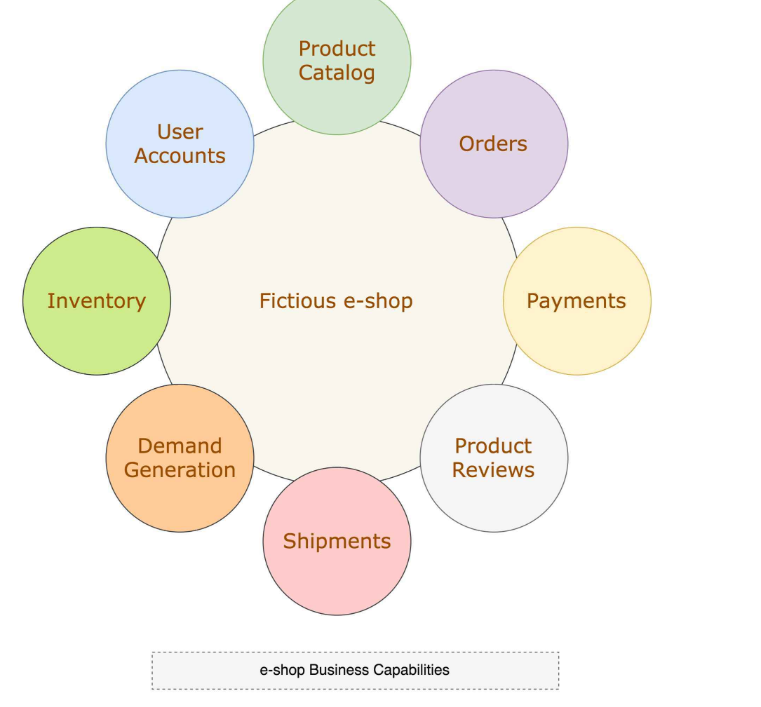
1. Domain Driven Design : Models should not b shared across contexts.

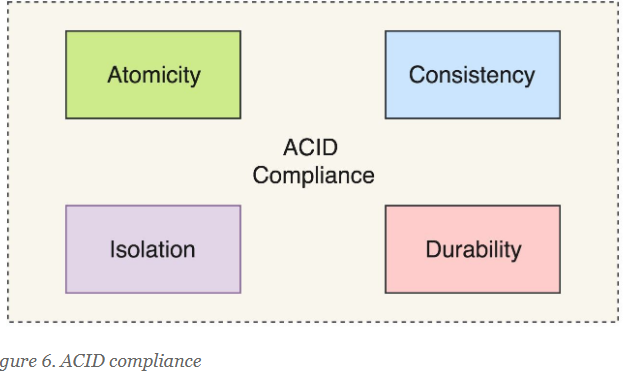
Polyglot Persistent: Multiple Technology

1. RDBMS: storing finanacial data
2. MongoDb: Product Catalog(Document oriented db)
3. Cassandra/Amazon Dynamo DB: Key value based DB(Analytics)
4. Redis: Distributed Caching, user session tracking
5. Neo4J: Graphical Db

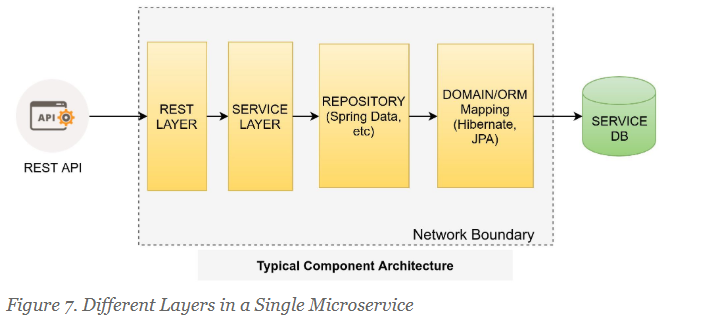


How to partition Large Application into microservice

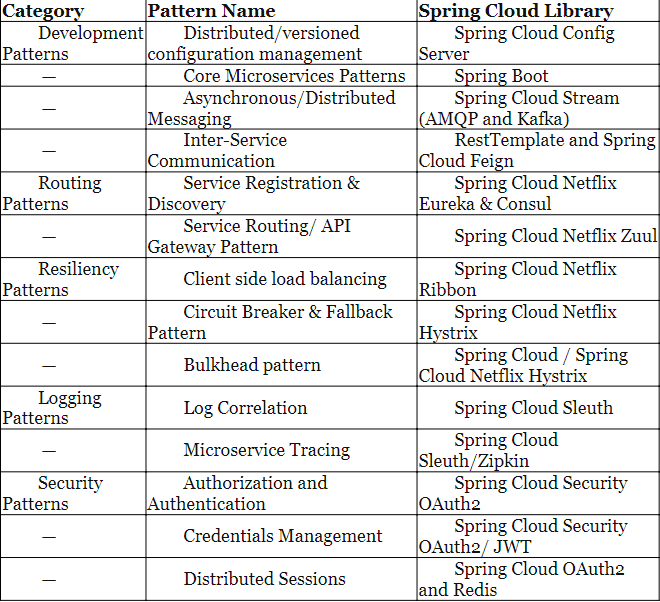




Different Layers of Microservice architecture:



Spring boot design pattern



|  |  |
| --- | --- |
|  |  |
| Different ways to create spring boot application | 1. Spring Initializr 2. Spring Cli 3. Adding Maven dependency |
| Used for REST calls | REST Template  Web Client (Reactive programming) |
| Discovery Server:  To eliminate url hard coding. |  |
|  |  |
|  | Eureka can also acts like a client :if you don’t want that then make this property as false  eureka.client.register-with-eureka=false |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |