Agenda

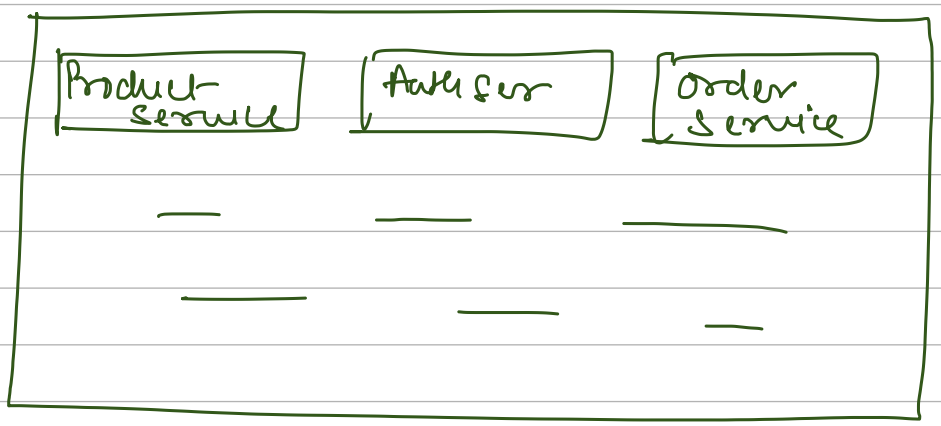
1) Project Overview
2) Microservice vs Monolith
3) Framework
4) Intro Spring framework
5) Dependency injection & IOC
6) Spring boot
7) first springboot Application

=> Backend of E commerce Service

- UserService
- SearchService
- InventoryService
- OrderService
- PaymentService
- LogisticsService
- NotificationService

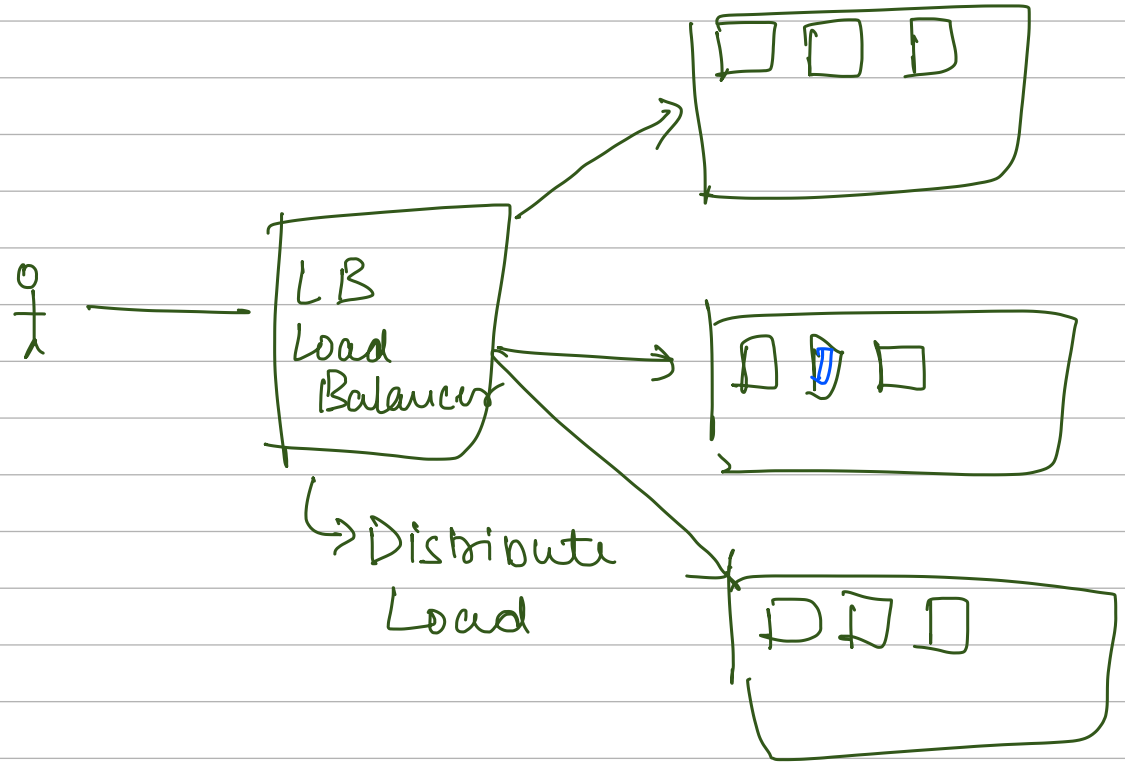
Monolith

MicroService



Amazon

We will have to run complete codebase on multiple server



Pros

1. Single Deployment
2. Easy Debugging
3. Low Latency

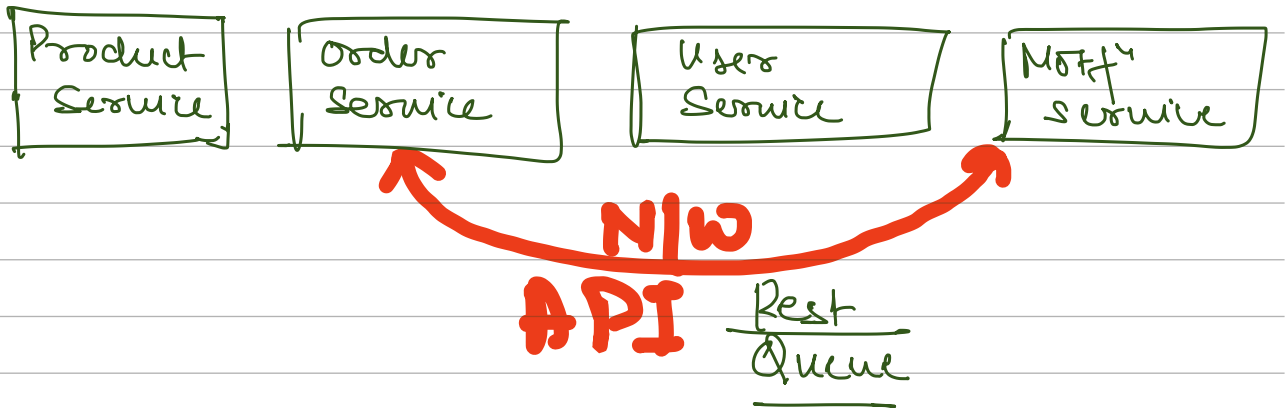
VIM

Cons

1. High deployment
2. No tech stack flexibility
3. No selective scaling
4. A small bug in one service can bring whole system Down

Scale of search = Scale for order

MicroService



Pros

1. Selective Scaling
2. Faster Development
3. Tech Stack flexibility

Cons

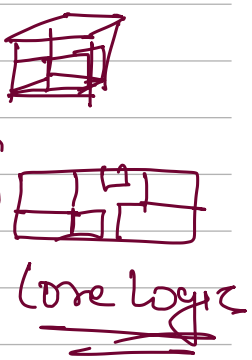
1. Difficult to debug
2. More latency

Micoservices

1) Product Service & Category Service
2) User Service
3) Search Service
4) Payment Service
5) Notification Service

Framework

Provides ready to use implementation of common things that many Software system will be using in their product



As a software engineer we want to spend more time in thinking about Bussines logic

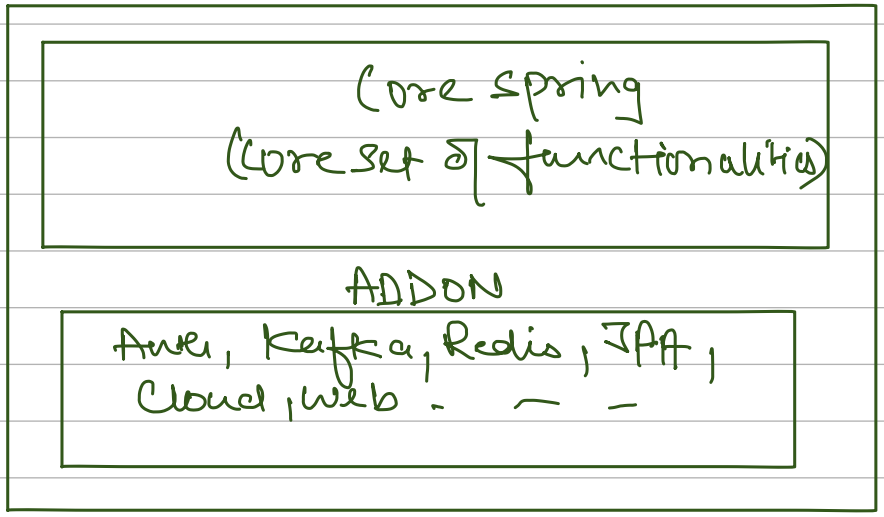
Framework provides functionalities in a ready to use manner for frequently occuring problems

- → Logging | Metrics
- → Talking to DB
- → Auth
- → Connecting Kafka, Redis etc
- → Web Application

Spring Framework (JAVA)

Set of projects that allows easy creation of Enterprise level JAVA application by allowing to do common things easily

(Production)



Springboot = Spring + Web

Dependency Injection

⇒ Class (A) is dependent on class (B)

1) Creating the dependency object within the class

```
Class A {
    B b = new B();
}
```

(11) Set the value of dependent object via Constructor / Setter

```
Class A {
    B b;
    setB(B b) {
        this.b = b;
    }
}
```

```
Class A {
    B b
    A(B b) {
        this.b = b;
    }
}
```

```
A a = new A();
a.setB( new B());
```

```
B b = new B()
A a = new A(b);
```

Creating dependencies outside of the class and injecting into the class is a better way of creating objects

① Reuse the same object

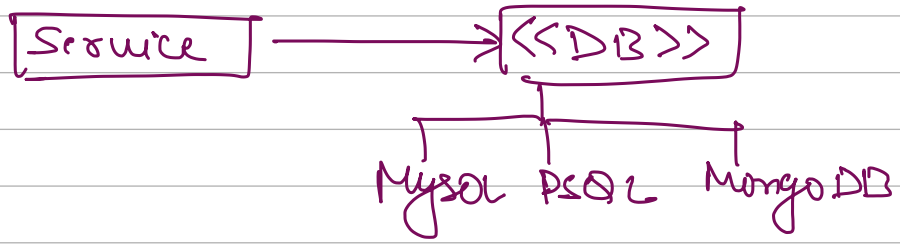
```
UserService {
    DB db;
}

ProductService {
    DB db;
}
```

Common DB object

```
DB db = new DB
US us = new US(db);
PS ps = new PS(db);
```

② Loose Coupling




```
UserService {
    DB db = new MySQLDB()
}
```

Phonpe → YES BANK × RBI Action

Phonepe

YESBANK YB = new YESBANK()

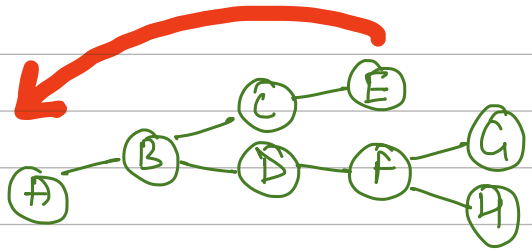
ICICIBANK

24 HOURS

YBBANK

ICICIBANK

BANK API = ( )



Topological

Spring : Dependency Injection

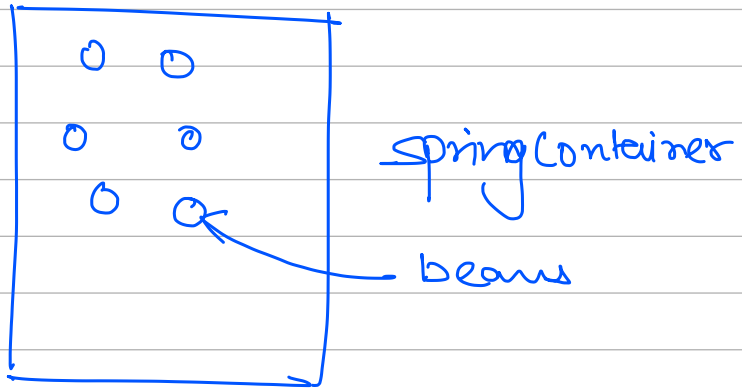
Inversion of Control

framework does the dependency injection on our behalf

Bean — Object

↳ Objects created by spring and use them for injection wherever required

Spring manages the complete lifecycle



1. Start the application
2. Create all the Beans
3. Store in Spring Container

1. XML Configuration
2. Annotation Based