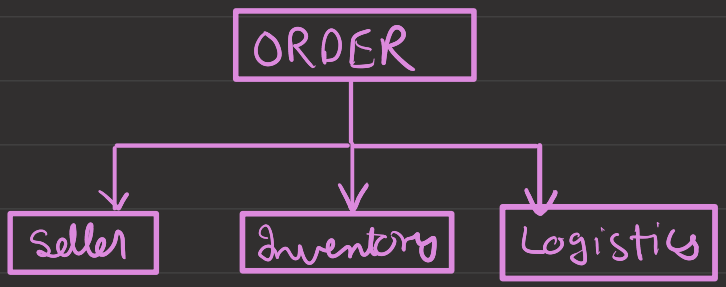
Designing Workflows in Microservices

Let's take an example to drive the topic

Suppose we are building an ecommerce platform. And there is an event of placing an order. And this in turn affects 3 other aspects:-

1. It needs to notify inventory
2. It needs to notify logistics
3. It needs to notify seller

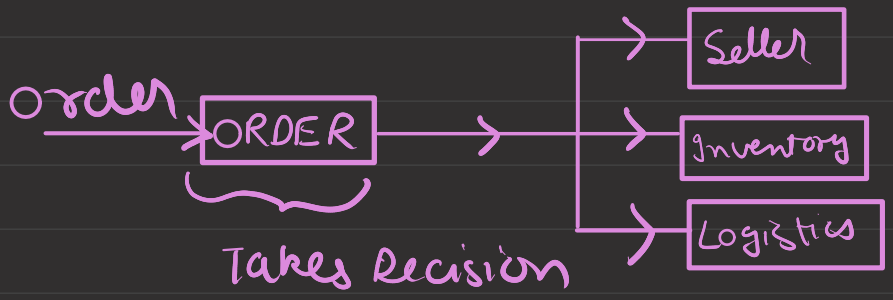
So on an higher level the order service is connected to 3 other services



To tackle this workflow, we have 2 main patterns

→ Orchestration — Decision logic is centralized

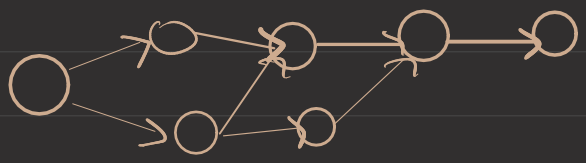
In this workflow we have the decision logic Centralized. A single brain to command.



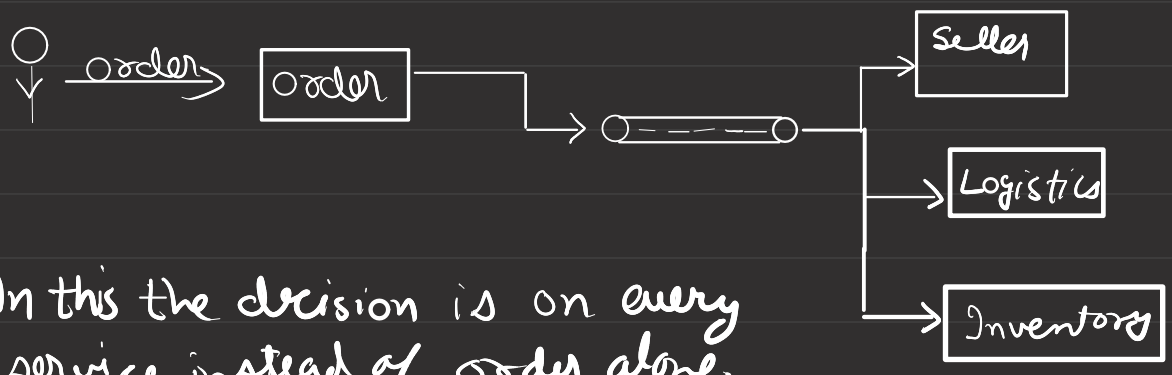
As soon as the order is placed, the order service makes the decision of sending data to the 3 services. The other 3 are basically "DUMB" as they just can't take decisions.

Basically one service "orchestrates" the work flow.

But as the complexity grows, it might happen that this one service gets overwhelmed



→ Choreography



In this the decision is on every service instead of order alone.
Once the order/calling service broadcasts the message the services can take up based on their need.

With this we can have better extensibility as we just need to plug a new service.

The order/calling service need not to manage the workflow alone

Loose coupling is also an advantage

Which one to use when?

Most modern services are inclined towards choreography

But But But

With choreography there is a thing of observability complexity. Your calling service does not know if the message has been used or not.

Because orchestration is REQ/RES type flow, we can use it at

- services need to be invoked transactionally - distributed transactions
- sending OTP for logging in happens synchronously
- rendering details in recommendation requires a sync communication