Agenda.

→ Communication b/w Microservices.

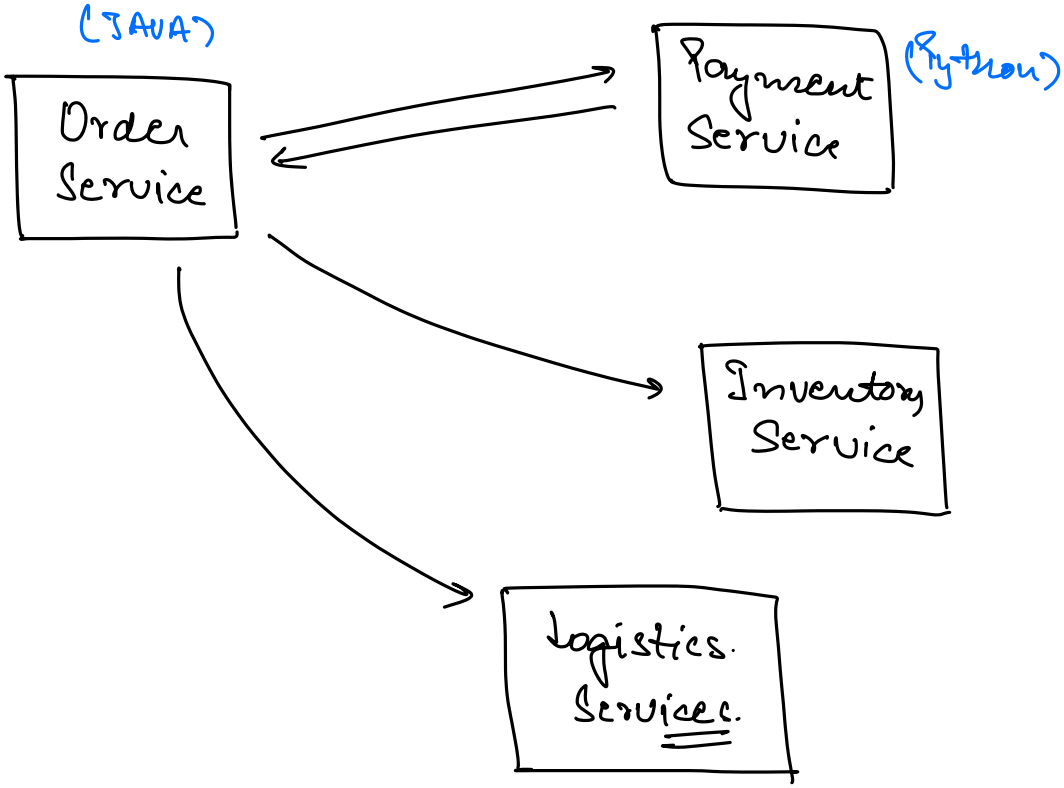
→ Consistency in Microservices.

- → 2 Phase Commit
- → SAGA.

→ CQRS

→ Circuit Breaker Pattern.

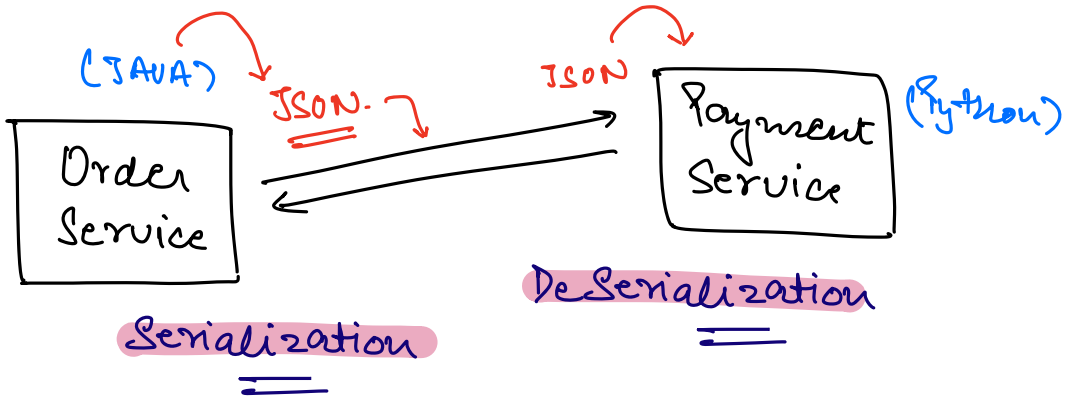
Communication b/w Microservices.



1) HTTP Request Response Model.

↳ API Call.

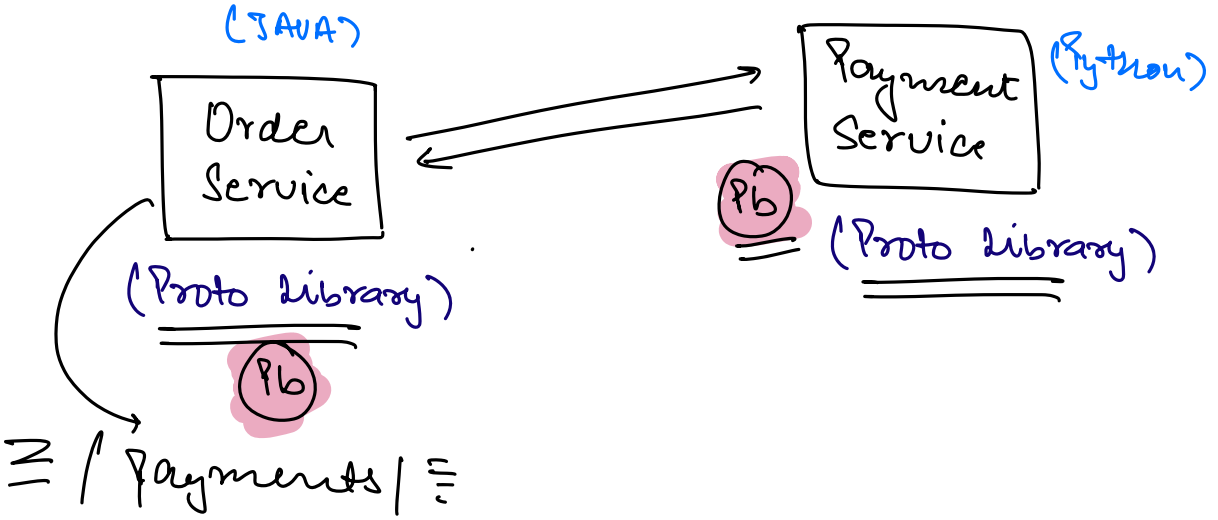
⇒ Synchronous



⇒ Takes time.

2) gRPC + Protobuf.
↓ ↳ Binary Data.

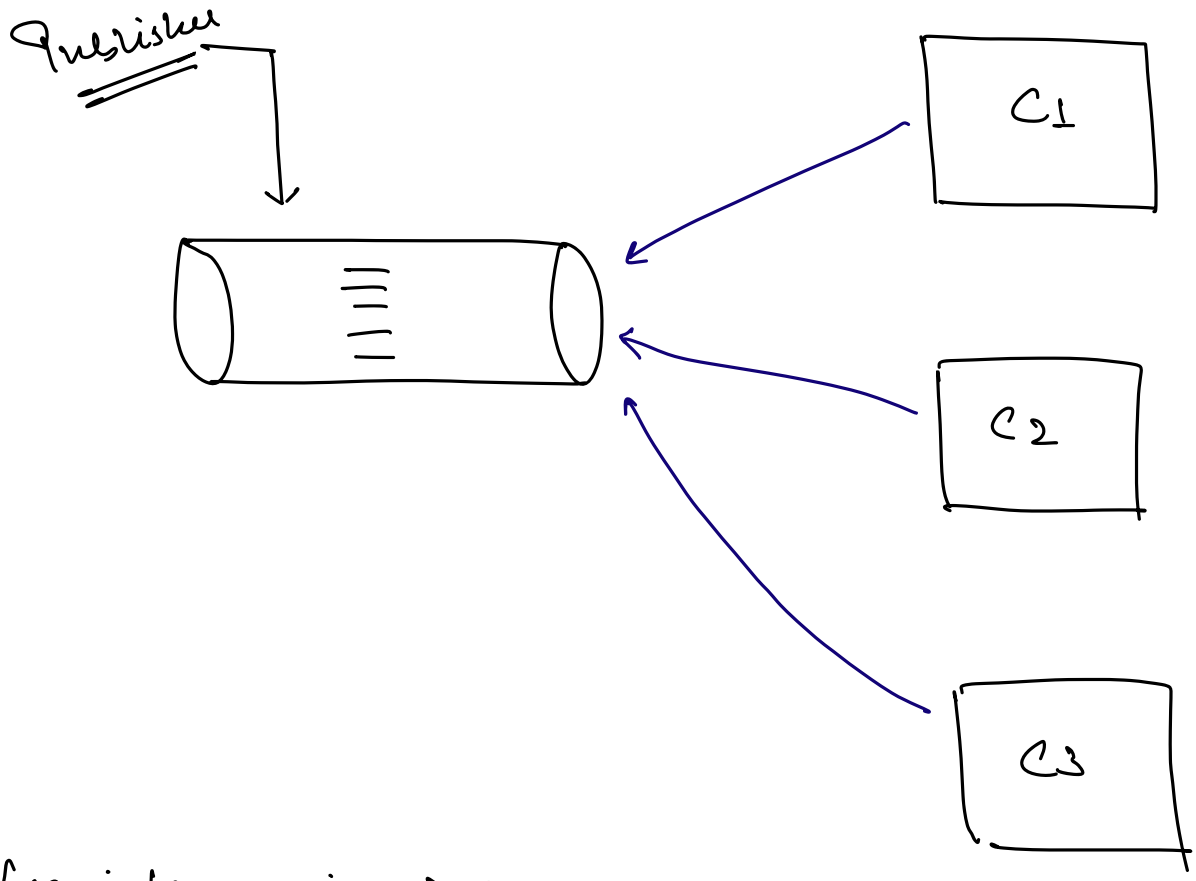
Google Remote Procedure Call.



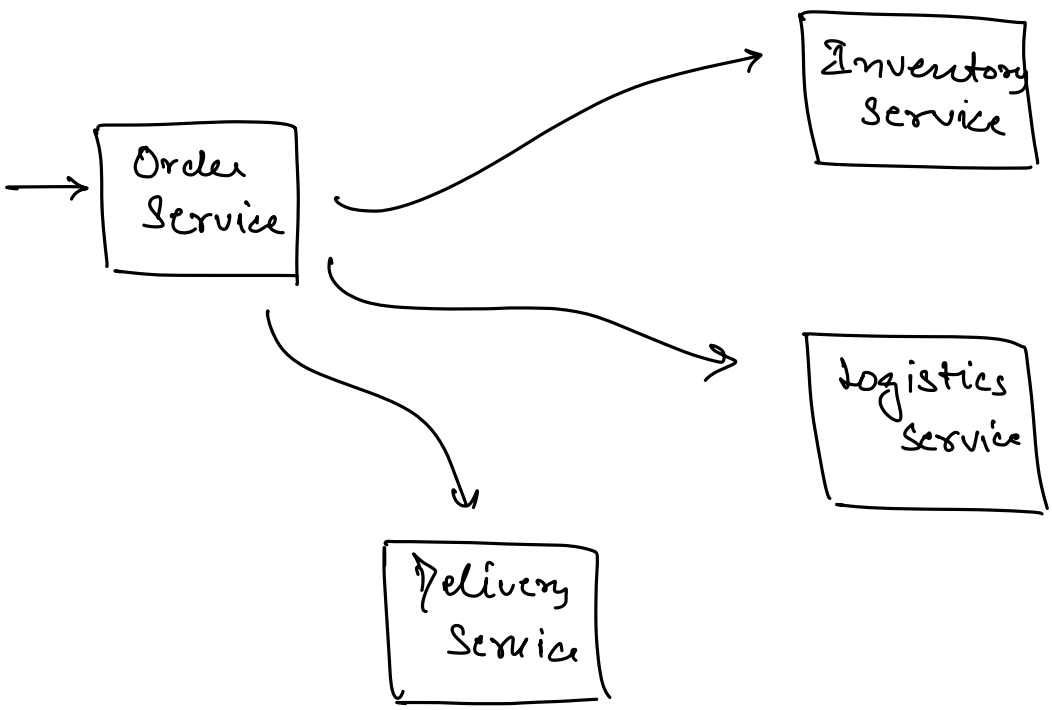
- placeOrder(java)
  ↓
  .proto

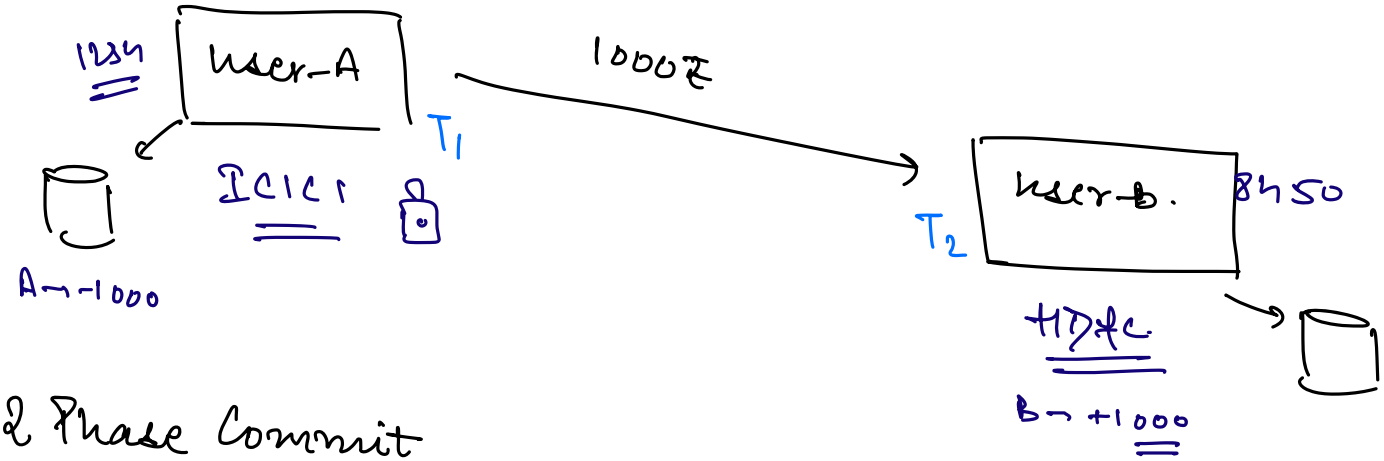
(11) Asynchronous | Event Driven Architecture

=> Hotstar | Netflix.



Consistency in Microservices.



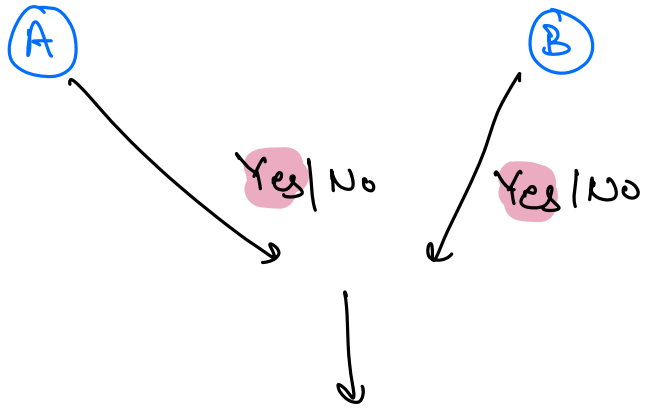


⇒ 2 Phase Commit

Phase I : Are you ready ? / Voting Phase?

⇒ Takes a lock on both the entries.

Phase II : Commit

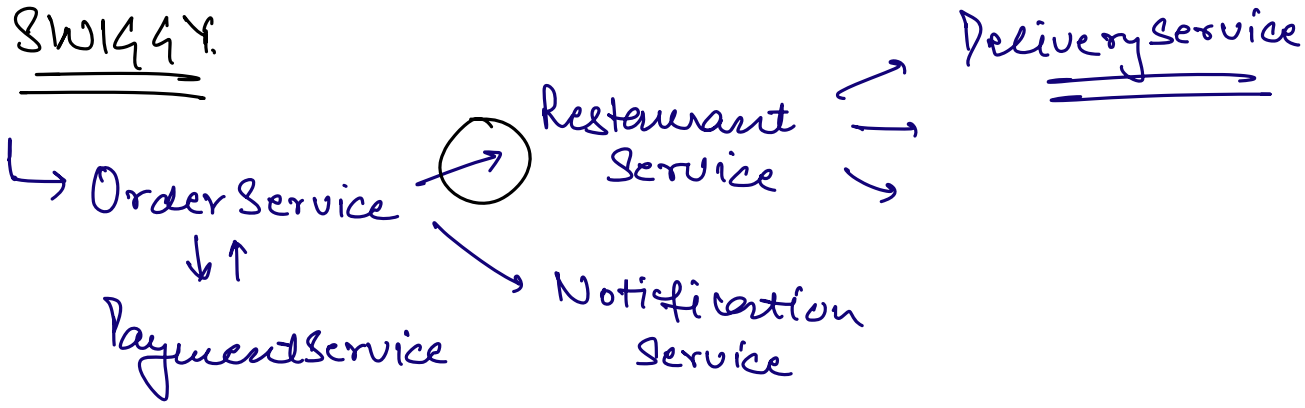


Commit, else rollback.

⇒ Highest level of consistency.

⇒ High latency.

⇒ SWIGGY.



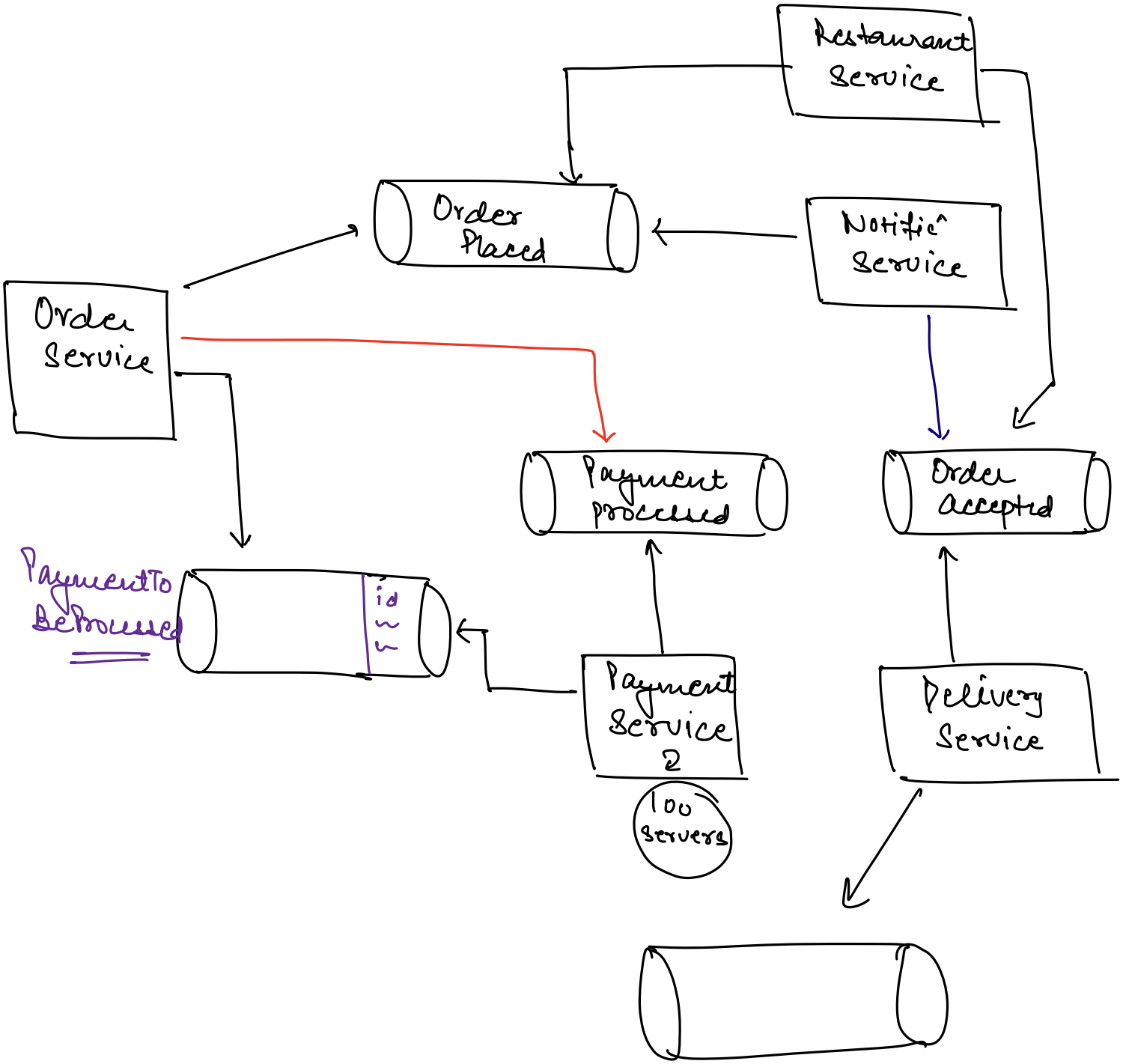
SAGA Pattern.

⇒ Orderly Chaos.

⇒ Different services are going to talk to each other via Message Queues.

⇒ Services will be decoupled.

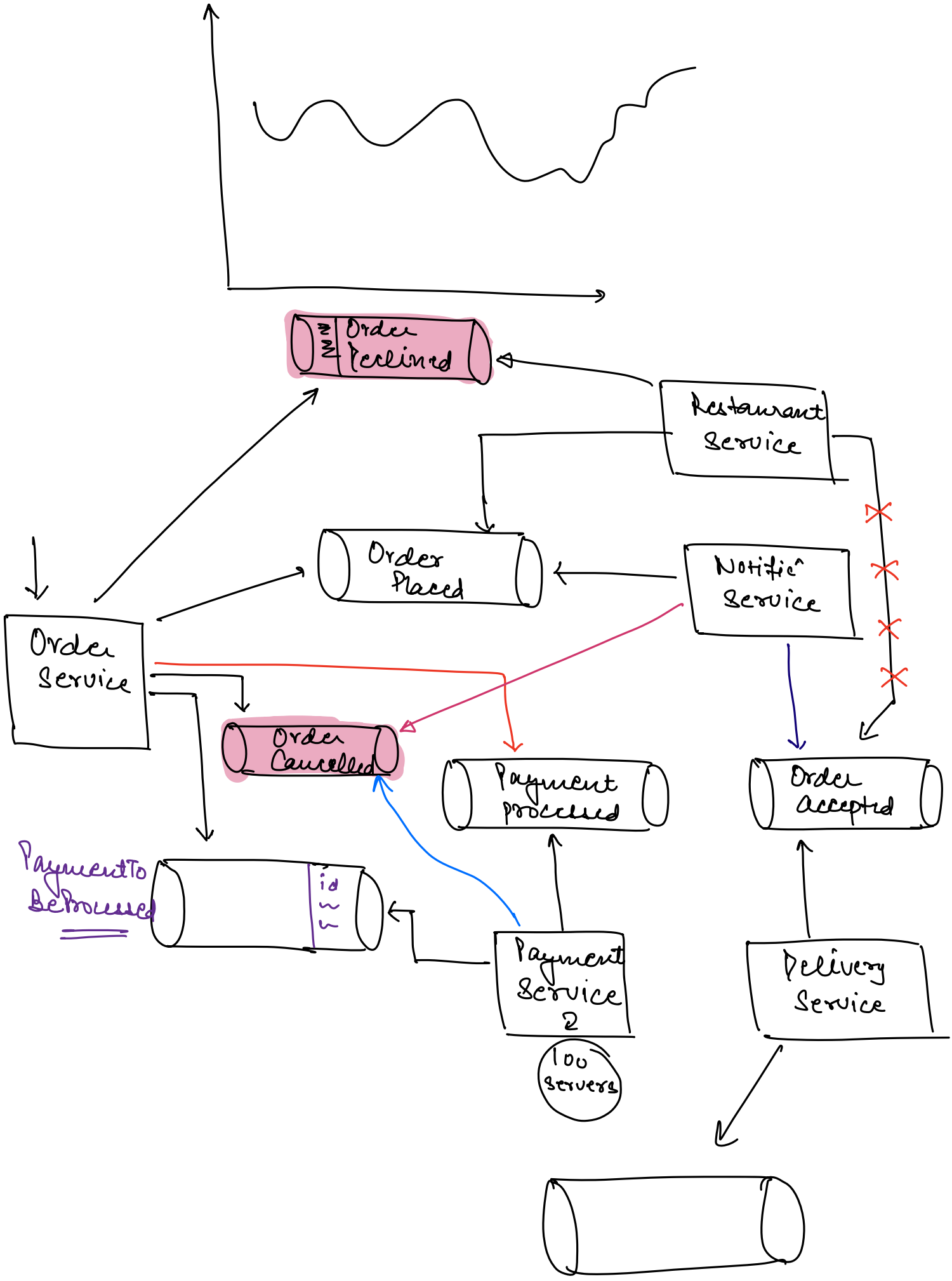
SWIGGY.



⇒ SLA : Service level Agreement.

⇒ Metrics.

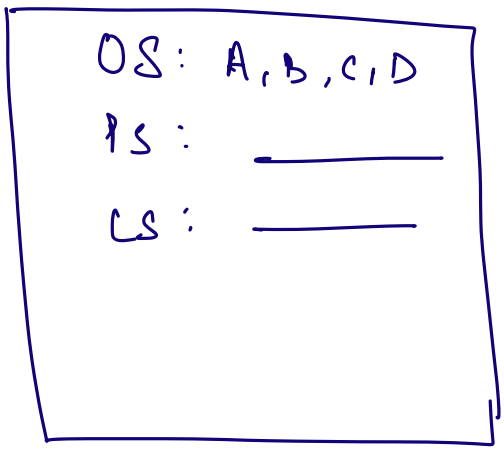
↳ Limits.


Order Declined
Restaurant Service
Order Placed
Notific^ Service
Order Service
Order Cancelled
Payment processed
Order accepted
Payment To be Processed
id
Payment Service 2
Delivery Service
100 servers

SAGA → Choreograph
SAGA → Orchestrated.

⇒ Service Discovery.

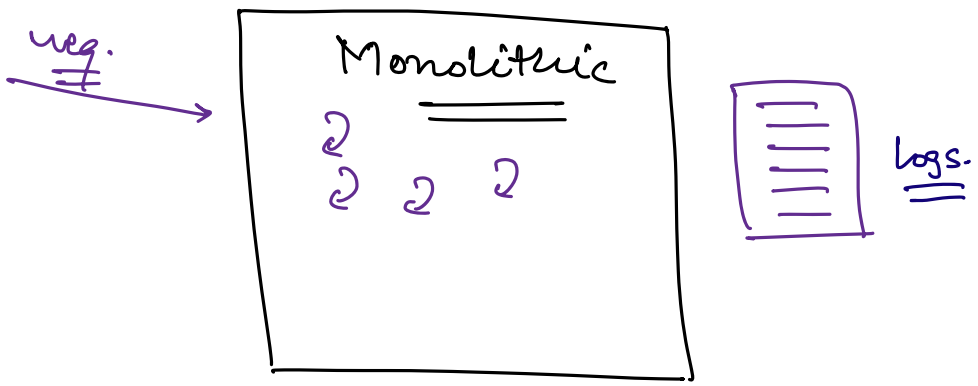
↳ Eureka Service.



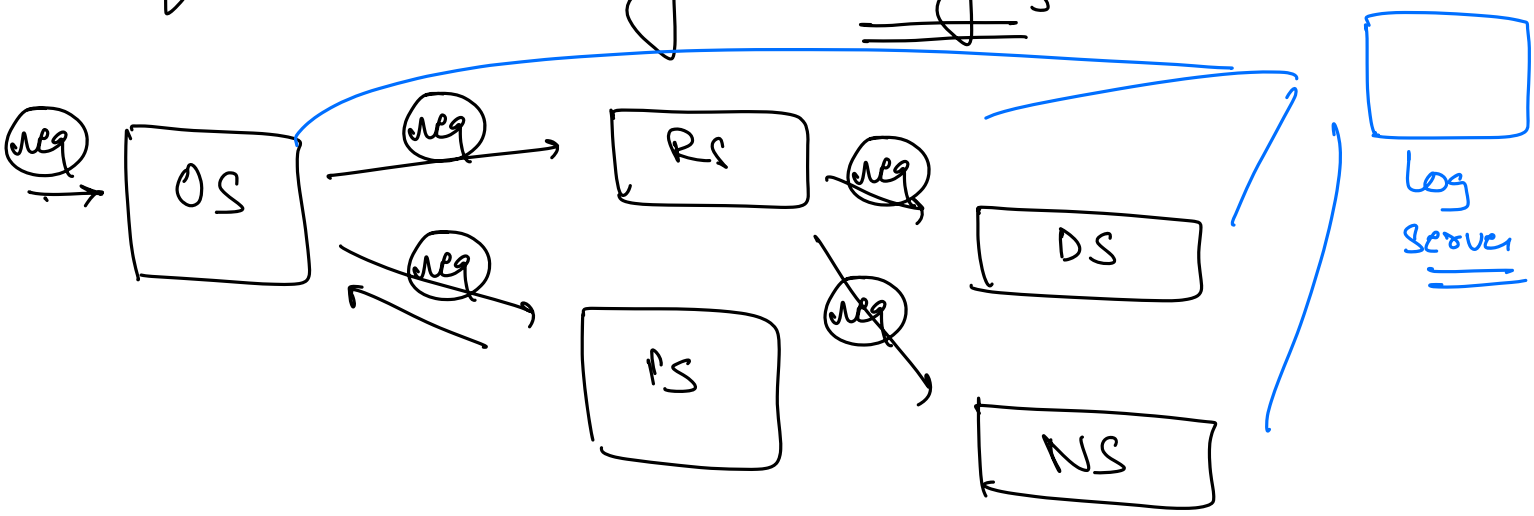
Observability in Microservices.

↓

Logging & Monitoring. → latency. → CPU → RAM



⇒ In Microservices architecture, tracking a particular request can be very challenging.



⇒ To debug an issue effectively, we need to keep a track of the entire request journey in right order.

⇒ Unique Trace Id | Correlation Id | tracking Id

Unique Id that is associated with the request at beginning itself & it stays with the request till the end.

AWS Cloudwatch

ELK → Kibana
Elastic Search → Index
Logstash