12/1/2024

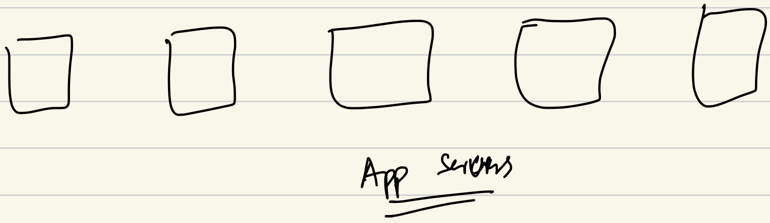
Microservice - 2

Flipkart

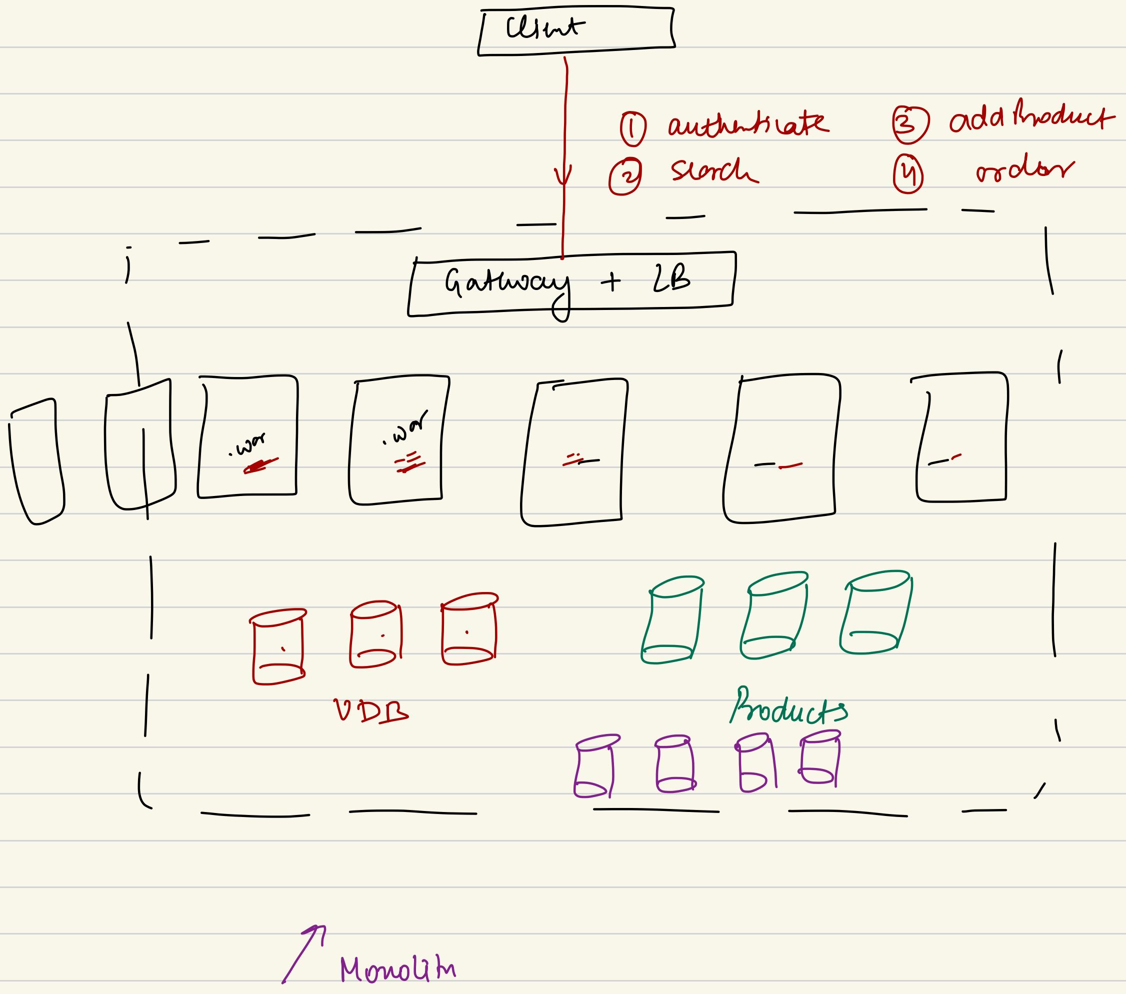
- Landing Page
- Search
- Authentication
- Product
- Order
- Payments
- Delivery

if all such functionalities are part of the same service

→ 1 executable (war | jar | . .)



Flipkart - Monolith



Problem:

① Tight coupling in code deployments; even a small code change in 1 logical use case will require us to re-build the entire code and redeploy on all machines.

② Given the tight coupling, we can't do targeted scaling of business use-cases.

③ Given the tight coupling, I can't select different tech stacks for different use-cases.

④ developer onboarding is going to be challenge

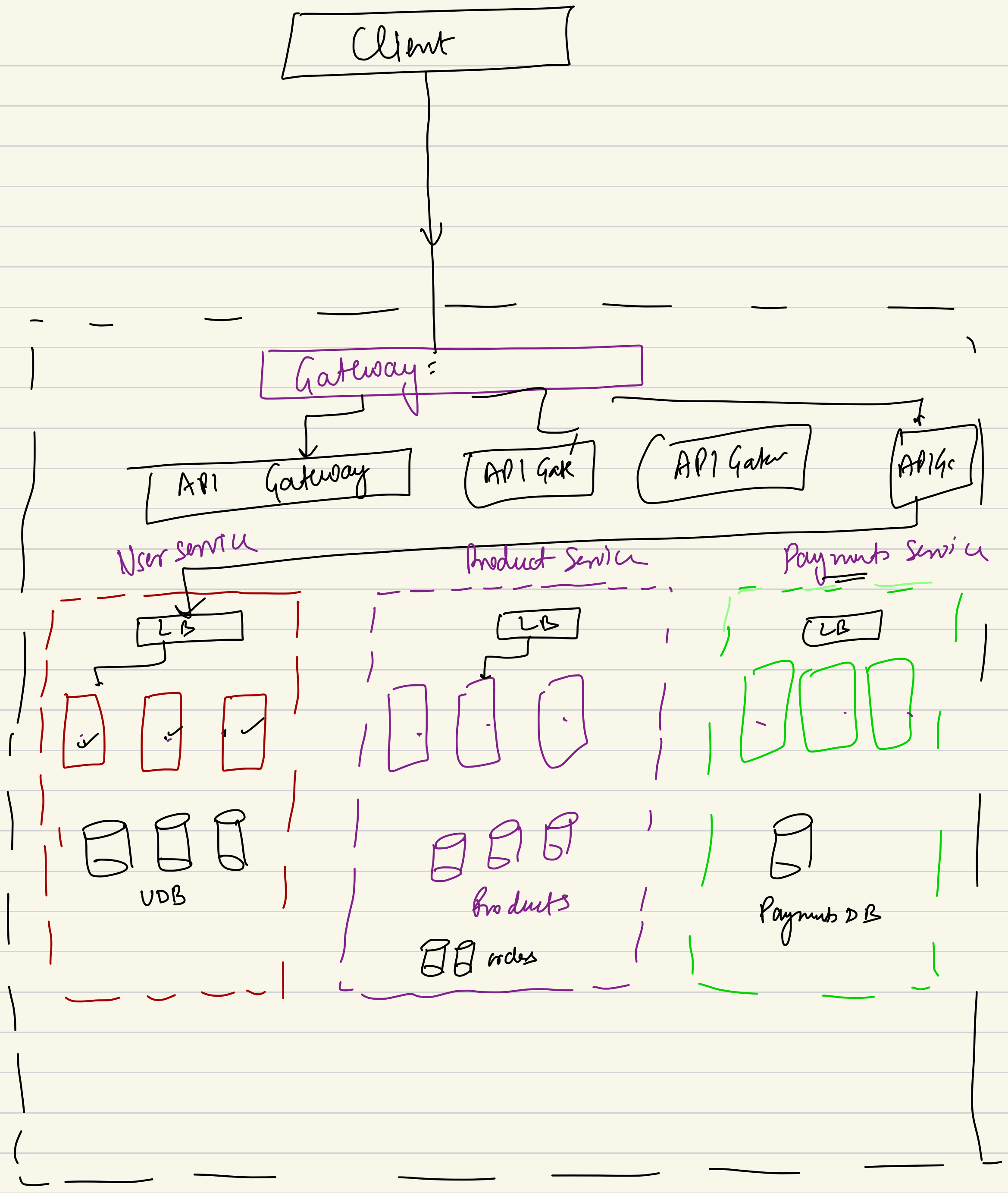
[Flipkart - Multiple Services]

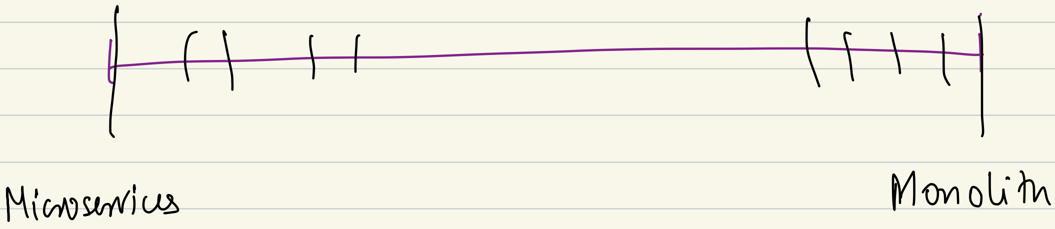
(I) User Service ✓

[sign-up
login
change password
change profile details]

(II) Product Service ✓

IV Payments service ✓


Client
Gateway:
API Gateway
API Gate
API Gate
APIG
User service
Product Service
Payments Service
LB
LB
LB
UDB
Products
orders
Payments DB



Disadvantages

① More # layers / hops ⇒ higher latency

② $$\begin{tikzcd}[row sep=small] & \text{more resources} \\ \text{Costly} \arrow[ur] \arrow[r] \arrow[dr] & \text{developer cost} \\ & \text{maintaining} \end{tikzcd}$$

③ $\boxed{\text{Logging + Monitoring}}$

④ (Interservice communication)

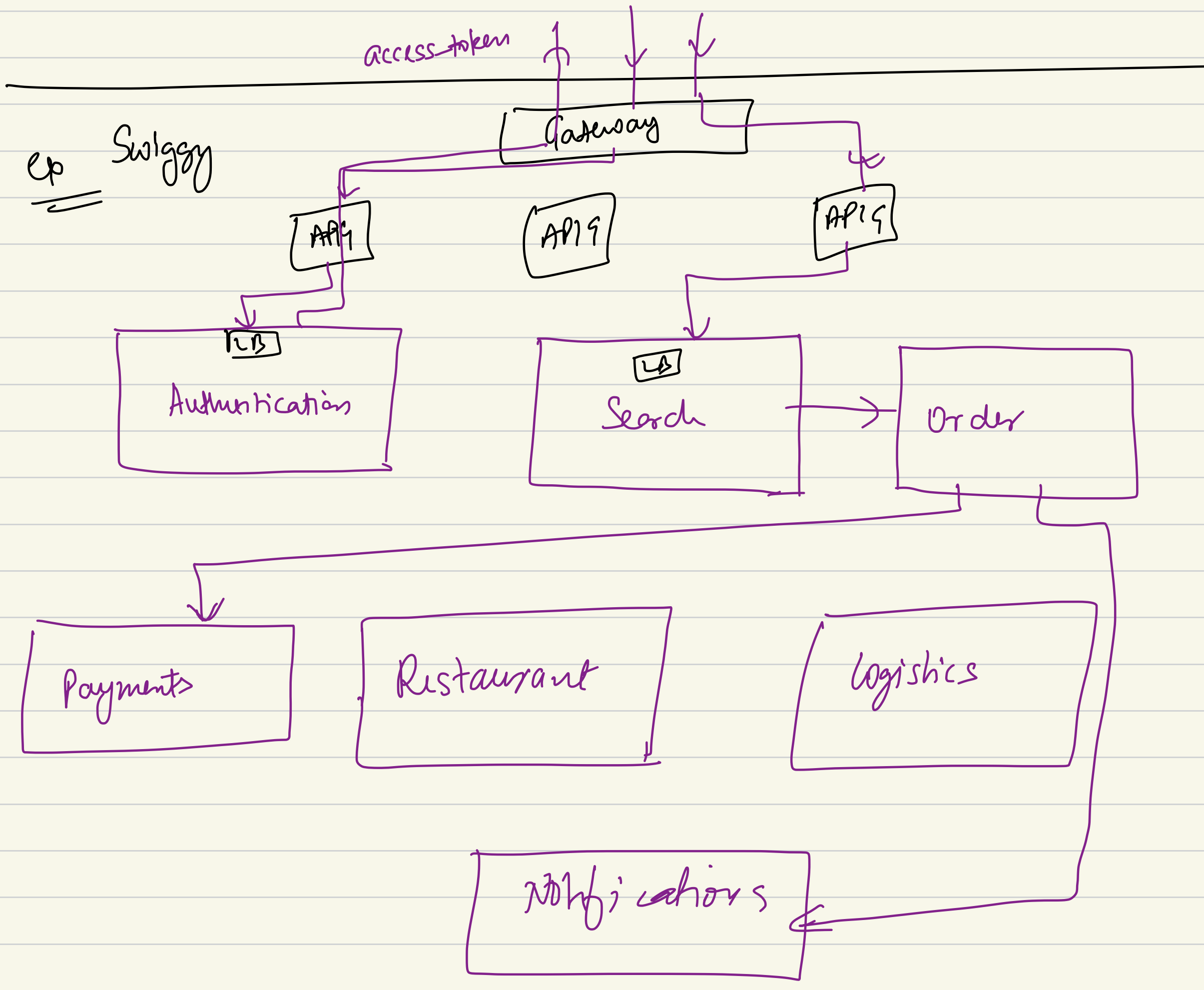
+

(Data inconsistency)

1. Communication b/w microservices

2. Observability
   - Logging
   - Monitor the metrics

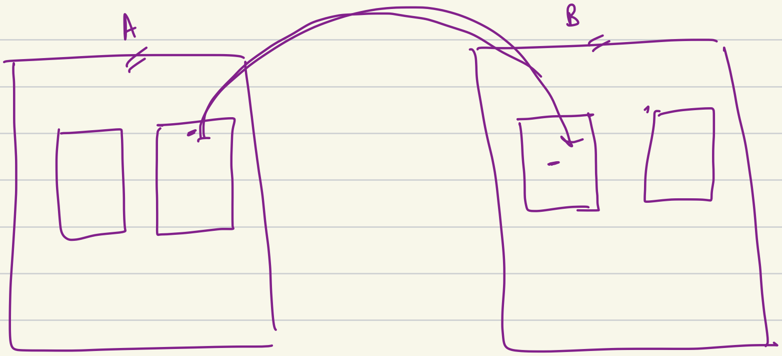
3. Transactions in Microservices
   - Data inconsistency



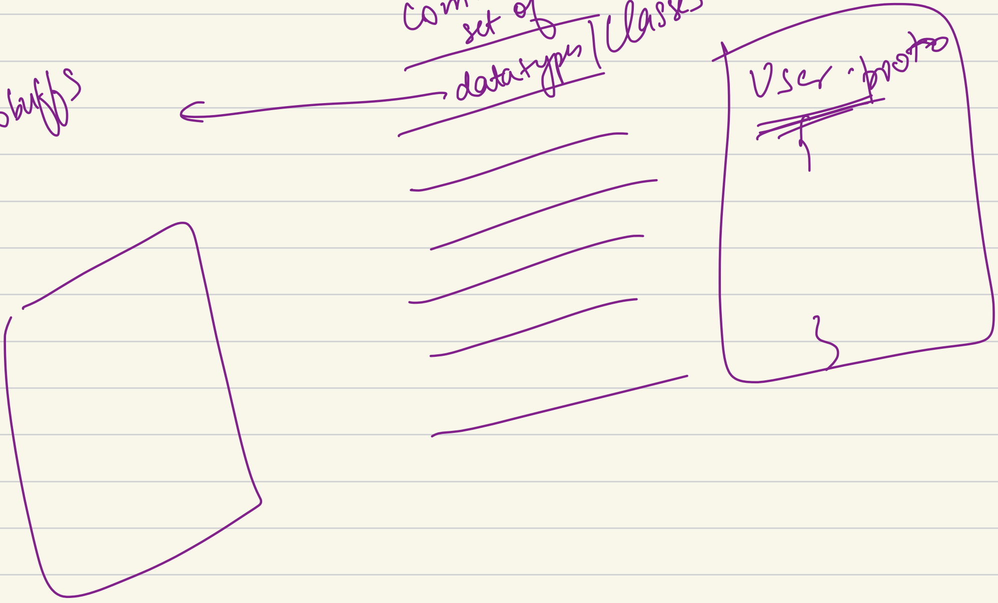
① ~~REST~~ API Call (synchronous)

② RPC (<u>Remote Procedure Call</u>) | gRPC (synchronous)

protobuffs



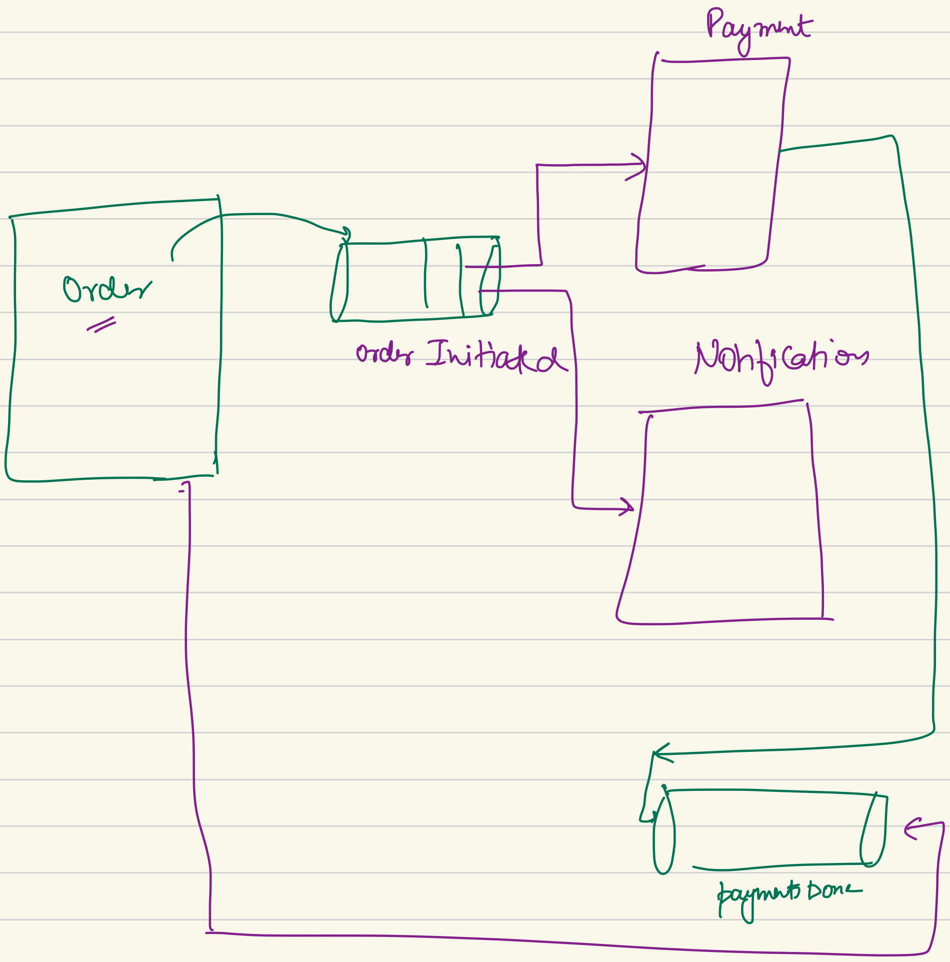
Protobuffs ← common set of datatypes | classes



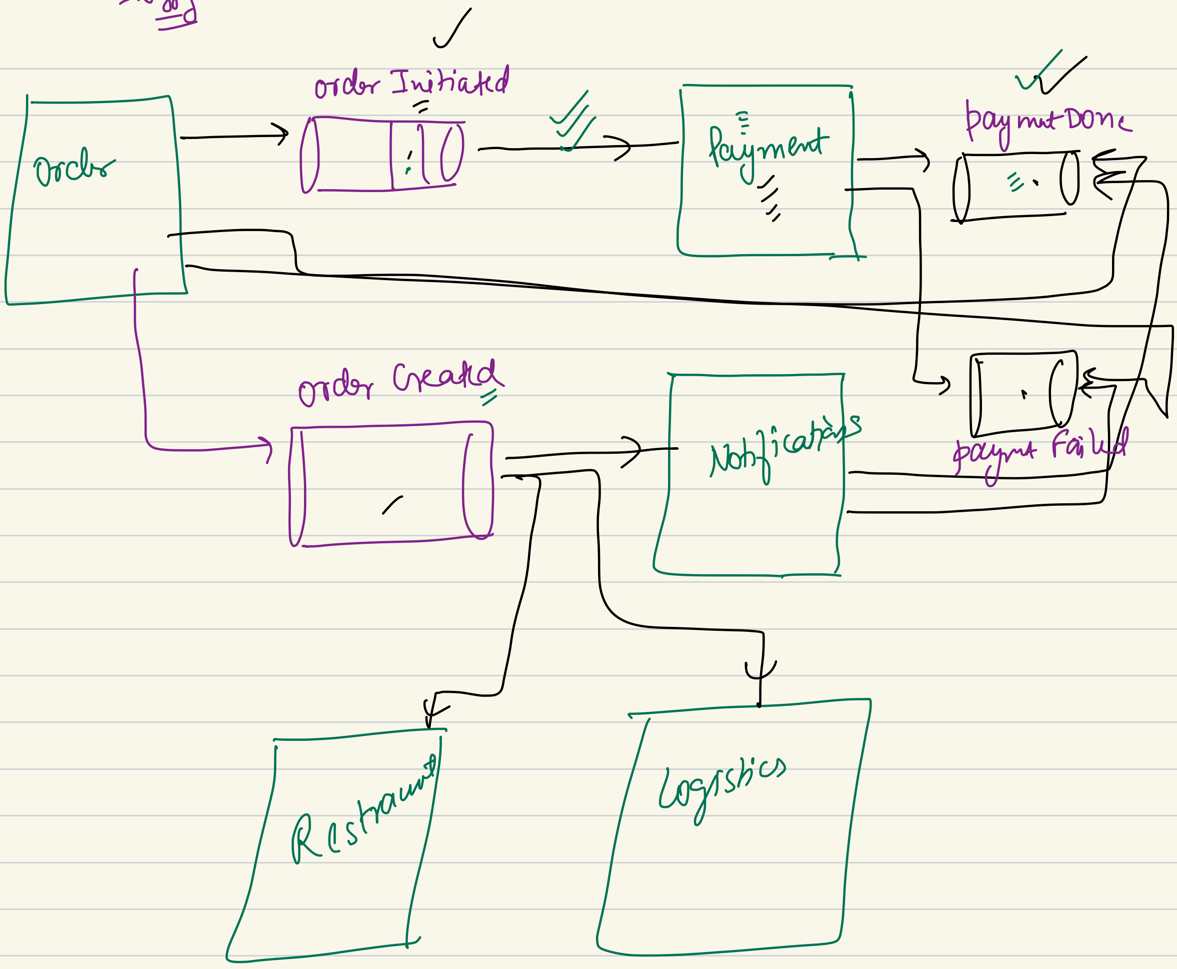
③ Message Queues [asynchronous]

— Kafka

Event Driven Architecture



Swiggy



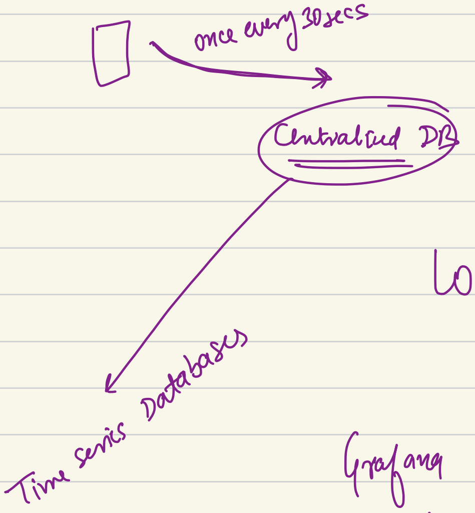
Event Driven Architecture

eg. Netflix

Observability

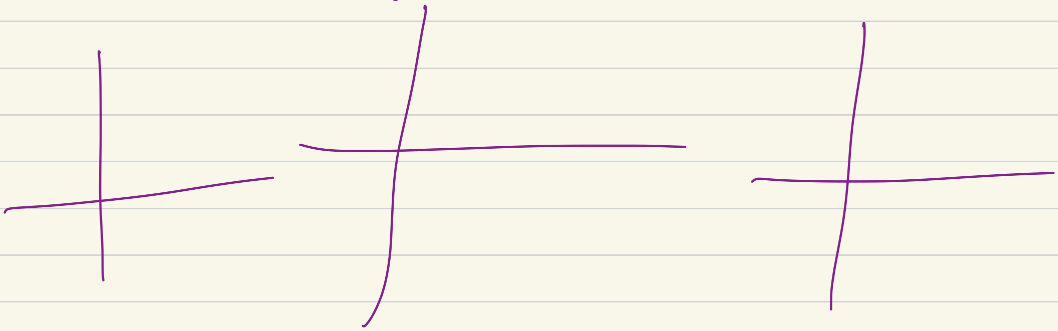
Metrics

CPU Utilizations, Memory Utilizations, latency, $\boxed{p95}$ latency, Avg Queue Size, % error, Database Storage %

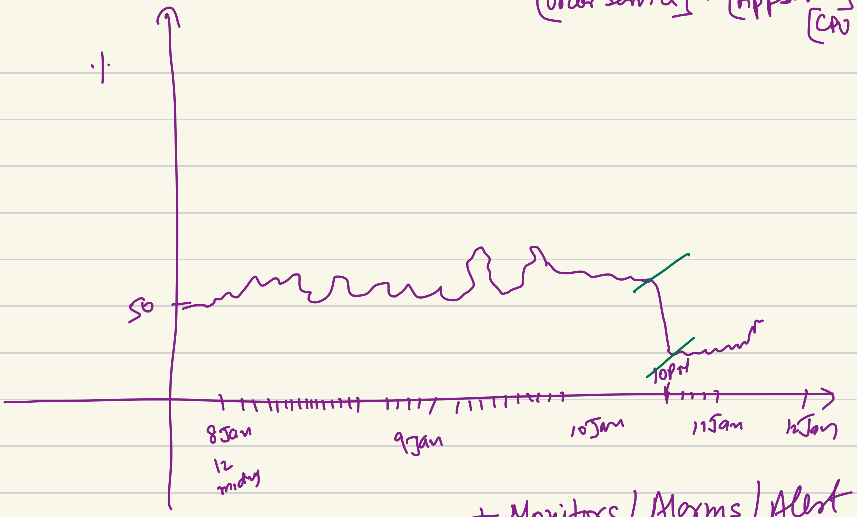


Logs

Grafana



[Order Service] — [AppServer] — [CPU Utilization]



+ Monitors / Alarms / Alert

eg AWS CloudWatch ☺

① Every machine across services

↓ send metrics at some periodicity

② Centralized DB ≡ Time Series DB

(similar wide coloumn DB)

↓ Plot Graphs on the aggregated values

③ Grafana

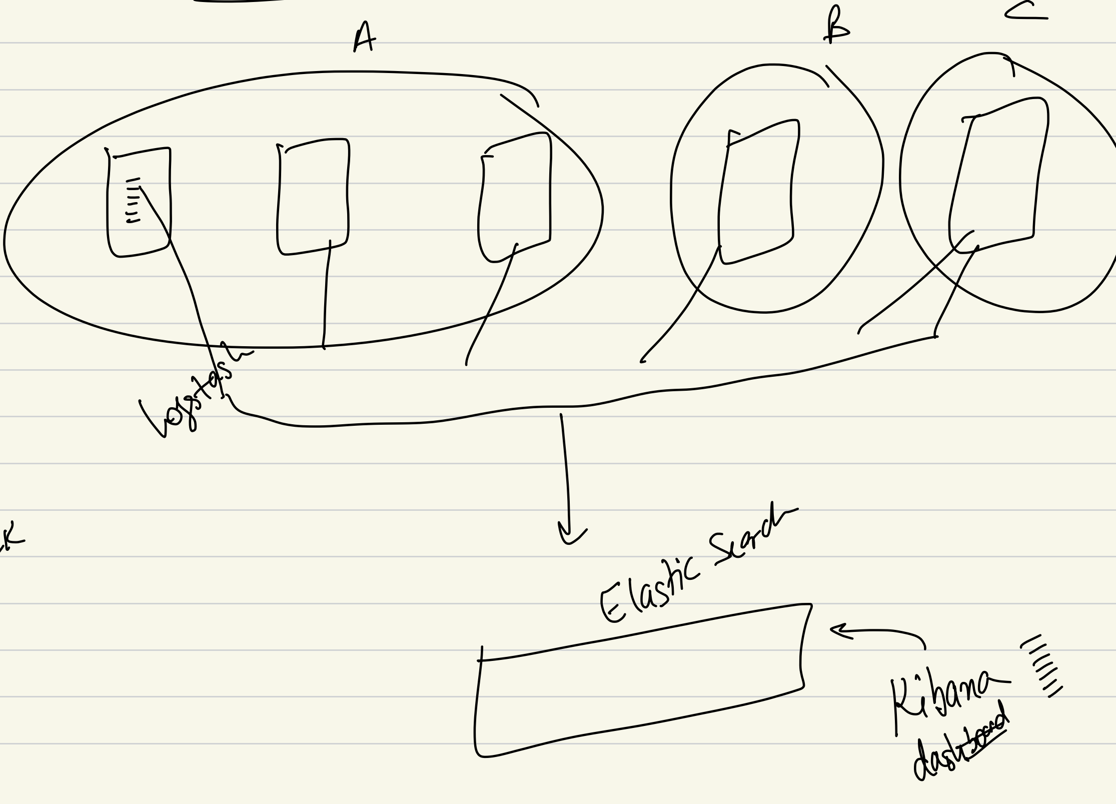
↓ Setup Alerts / Monitors

④ devOps

OBservability

$$\begin{tikzcd}
 & \text{OBservability} \arrow[dl] \arrow[dr] & \\
\text{(I) Monitoring | Metrics} & & \text{(II) Logs | Tracing}
\end{tikzcd}$$

Distributed Tracing



ELK

① Unique Trace Id gets passed along for the same request across all microservices

ex. We can generate a msgId and keep on passing it at every step

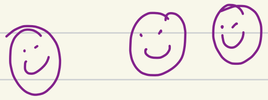
② If we have logs related to this unique trace Id everywhere, and we are able to collect all these logs at a central place, Then we can aggregate or search by this id

Elasticsearch

③ The stage/point where the logs are missing (if any), that is where error happened eg Splunk , Elastic Search

$$\begin{tikzcd}
 & \text{ELK} \arrow[dl, dash] \arrow[d] \arrow[r] & \text{Kibana} \\
\text{Elastic Search} & \text{Logstash} &
\end{tikzcd}$$

Metrics $\longrightarrow$ Hot Path



less latency

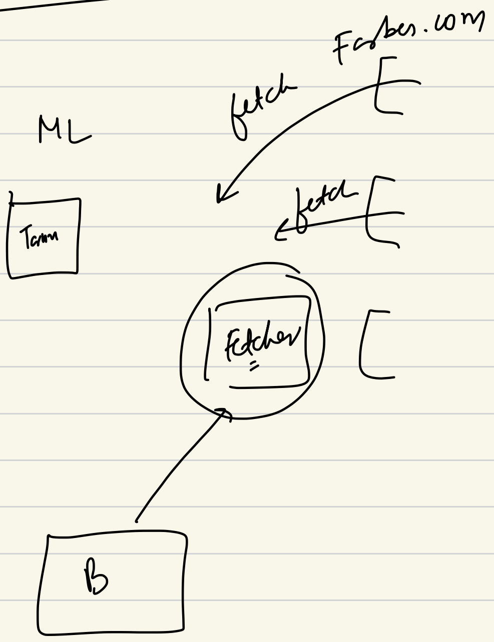
Logging $\rightarrow$ Warm Path

slightly more latency

Data Consistency

QUESTION

SOA



Forbes.com
ML
fetch
Team
fetch
Fetcher
B