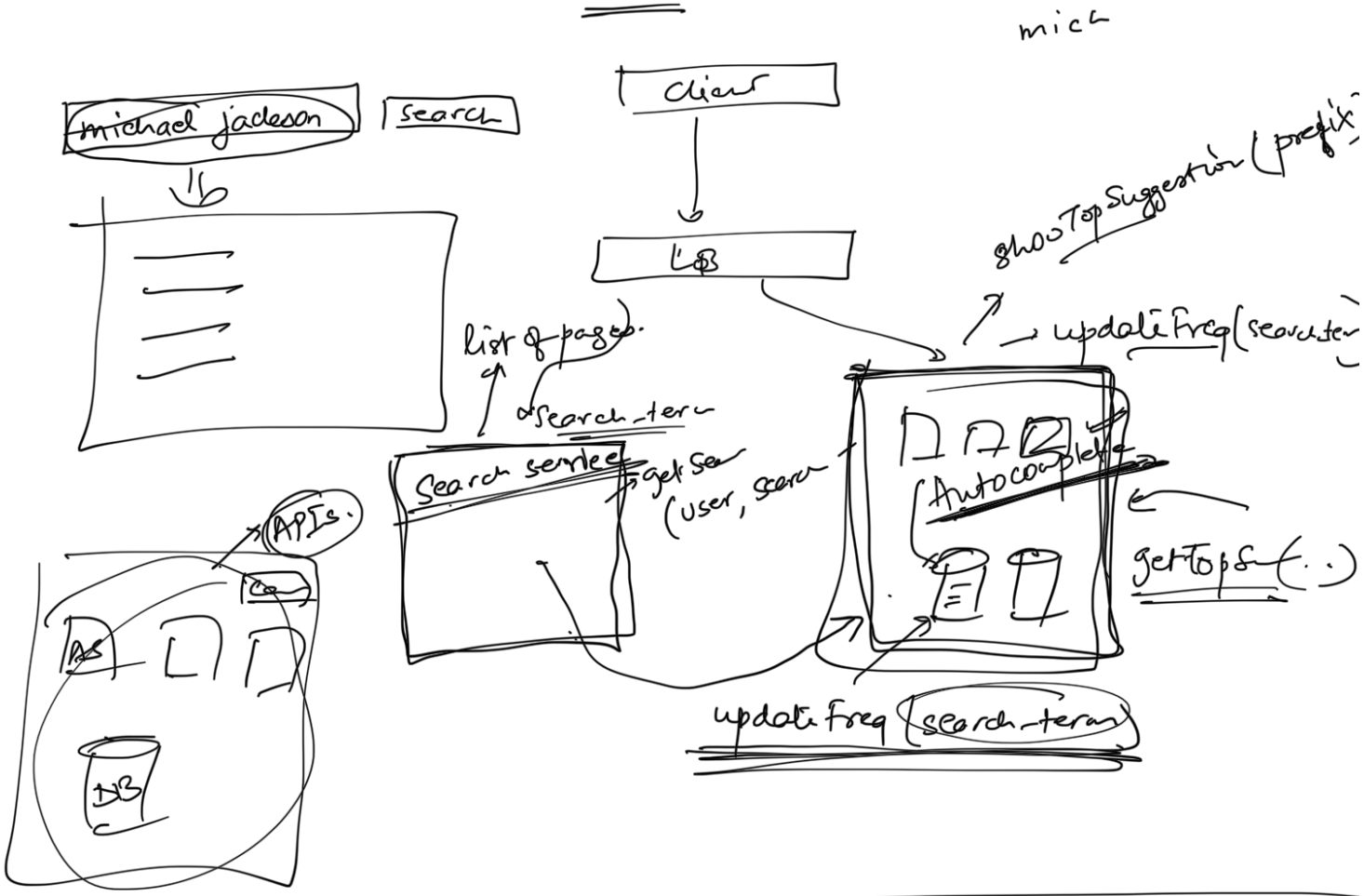
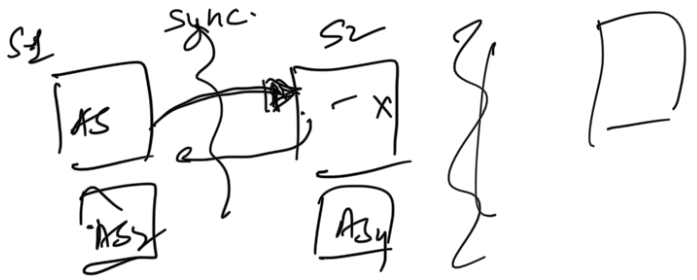
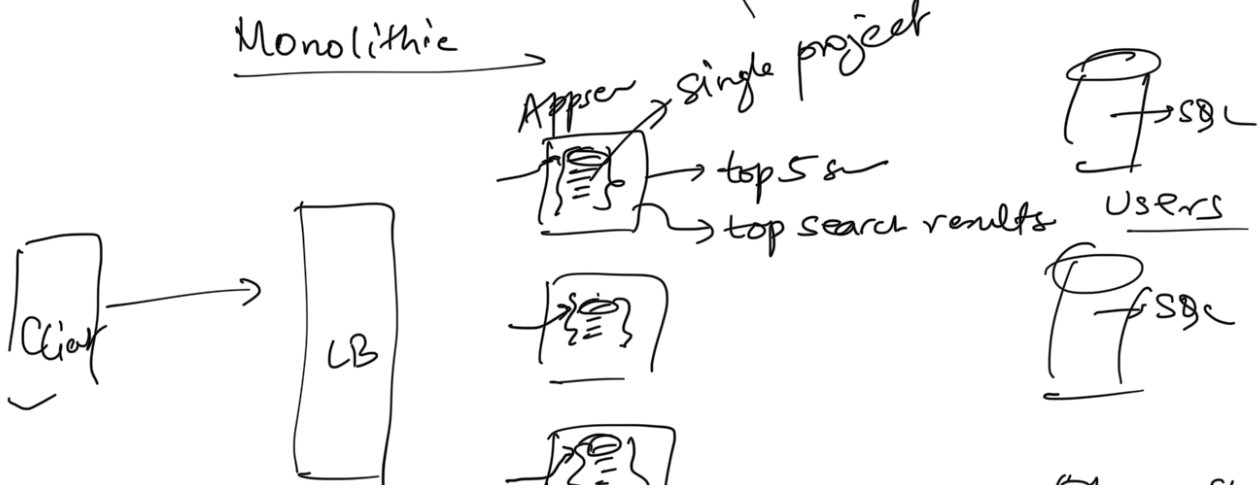
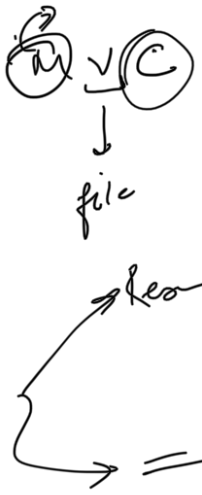
MICROSERVICES

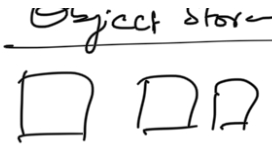


Service Oriented Architecture	Microservices
	① Separation of concern ↳ duplicating data ② Event driven architecture

Monolithic



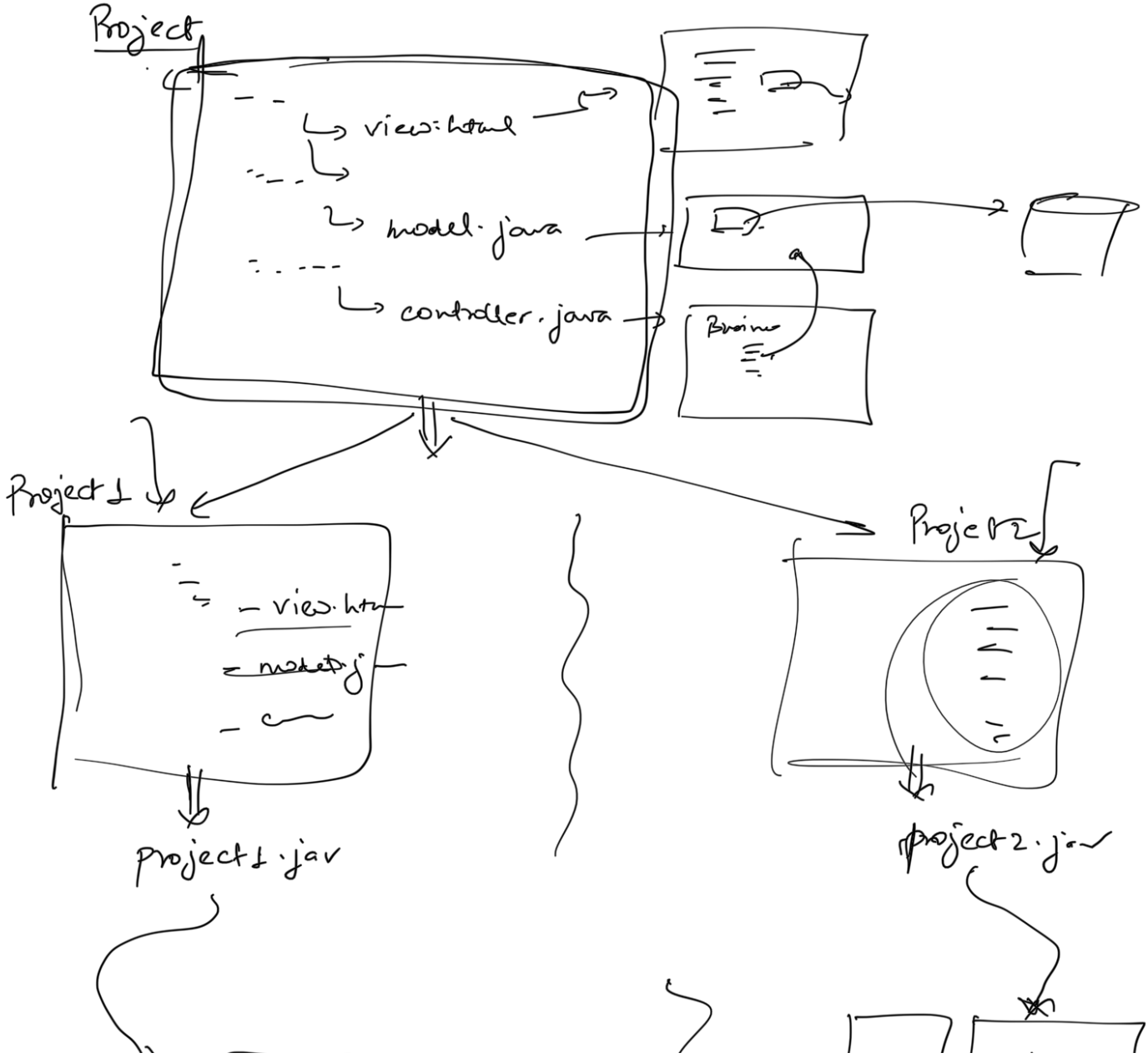


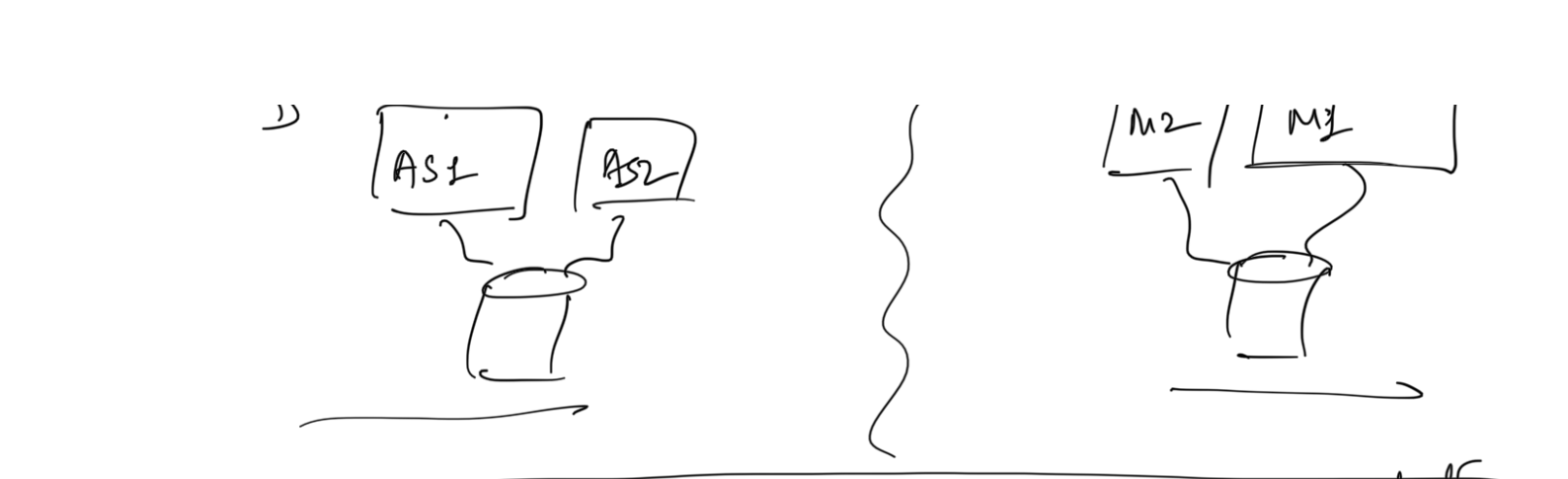


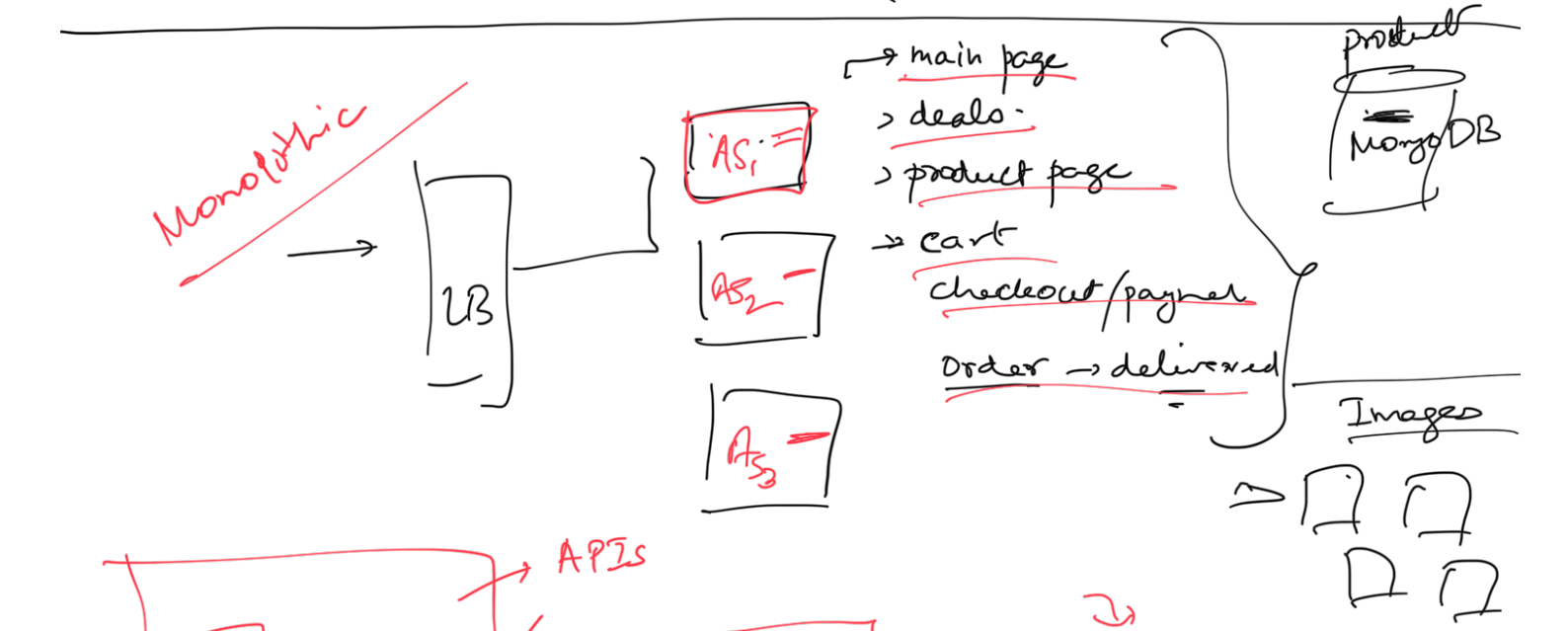
→ Coupling,

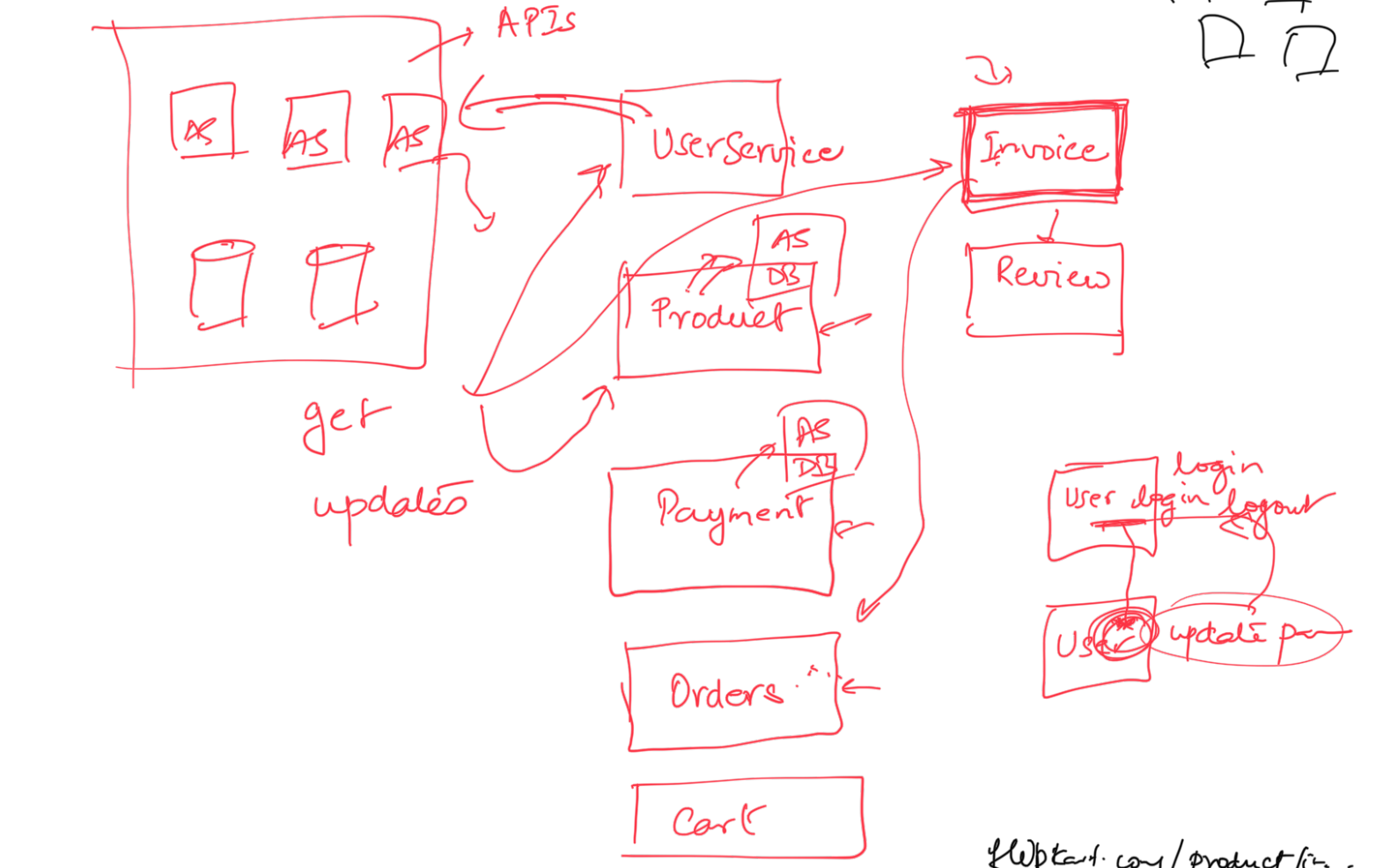
1. New employees/engineers.
2. All minor changes → large project compilation + deploy
3. Project compilation become slower
   → Build time
4. Failure/ slowness in one component causes everyone to be affected

---



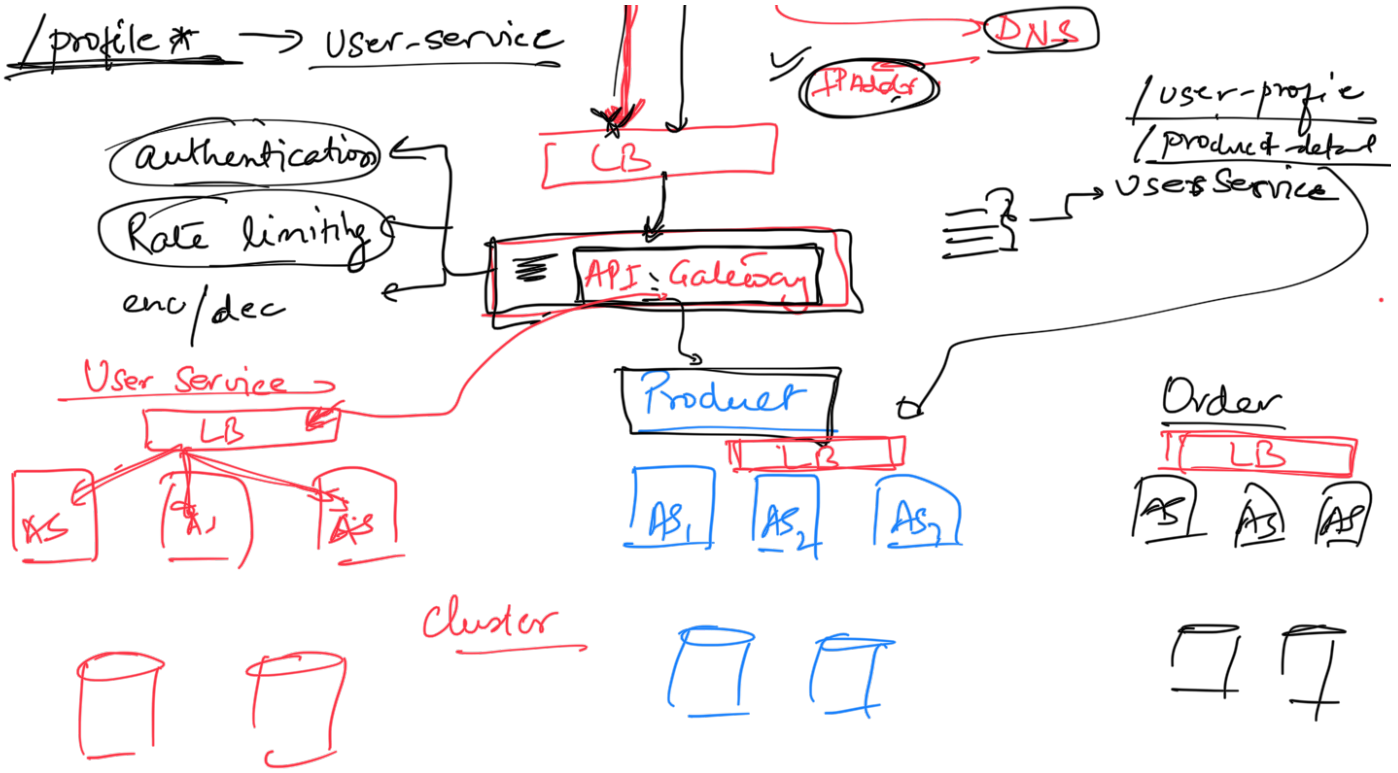


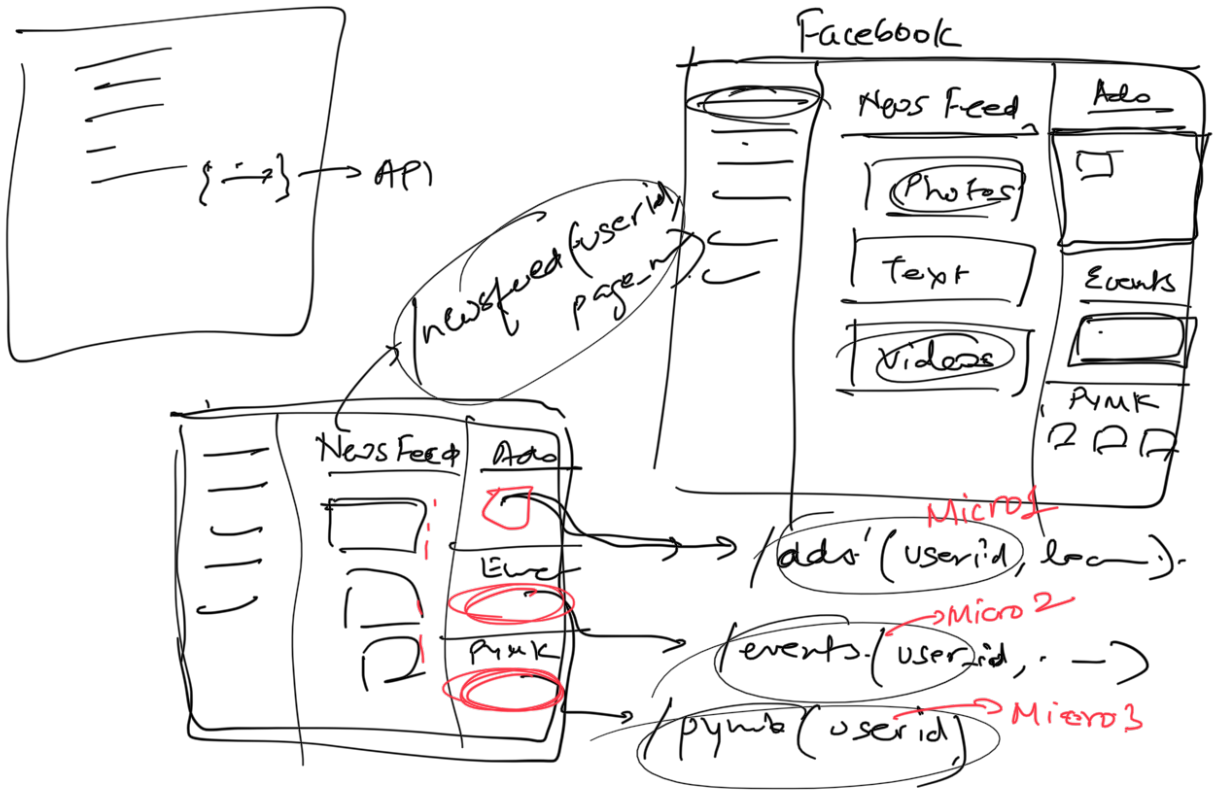


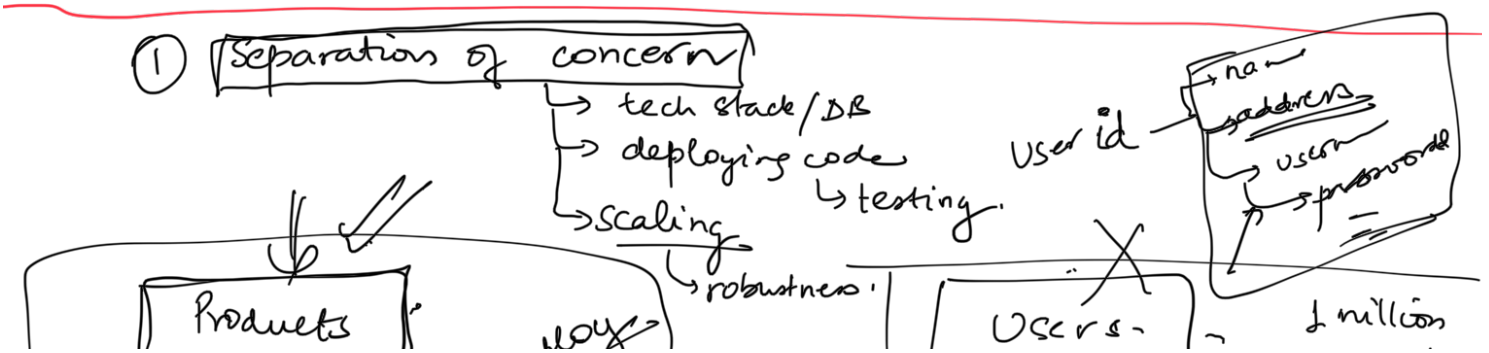


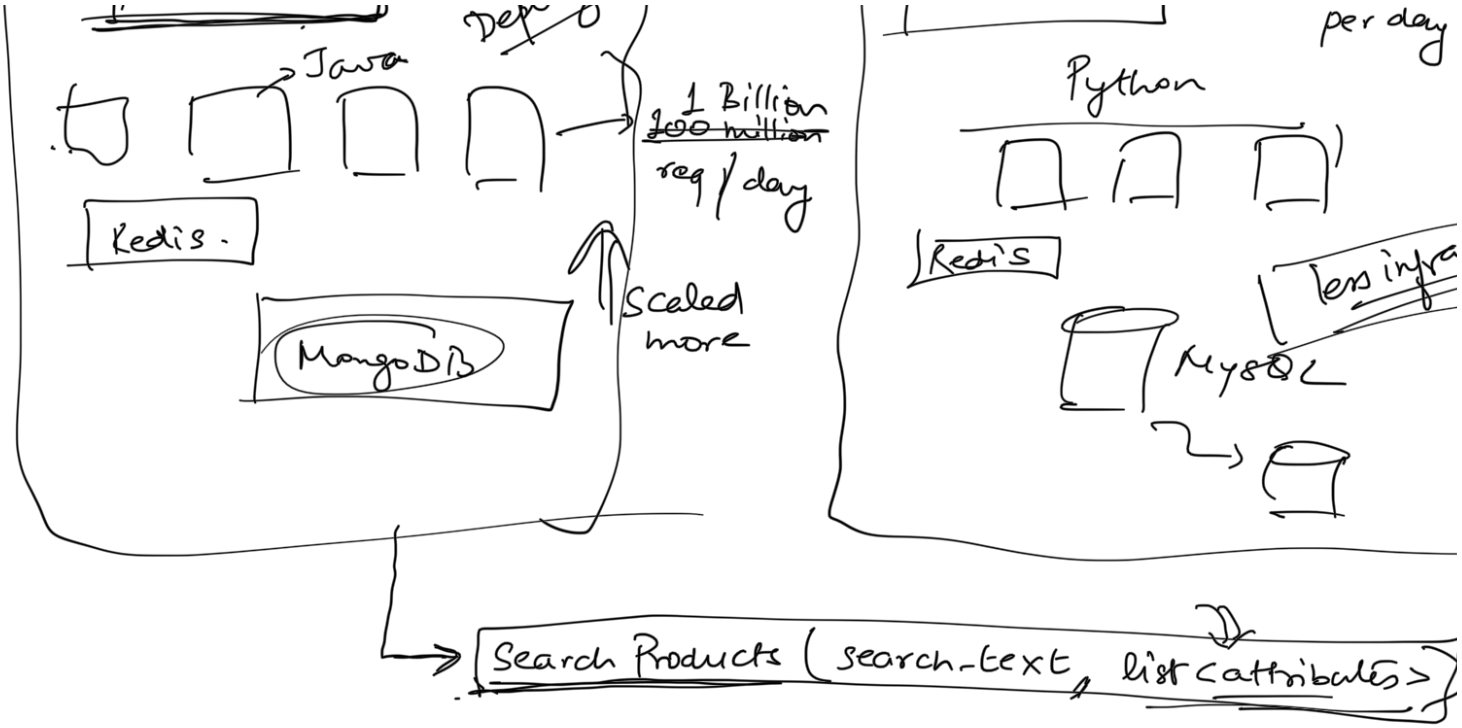
/endpoint → microservice

Client


/profile/* → User-service
DNS
IPAddr
/user-profile
/product-detail
Users Service
Authentication
Rate limiting
enc/dec
LB
API Gateway
User Service
LB
Product
LB
Order
LB
Cluster

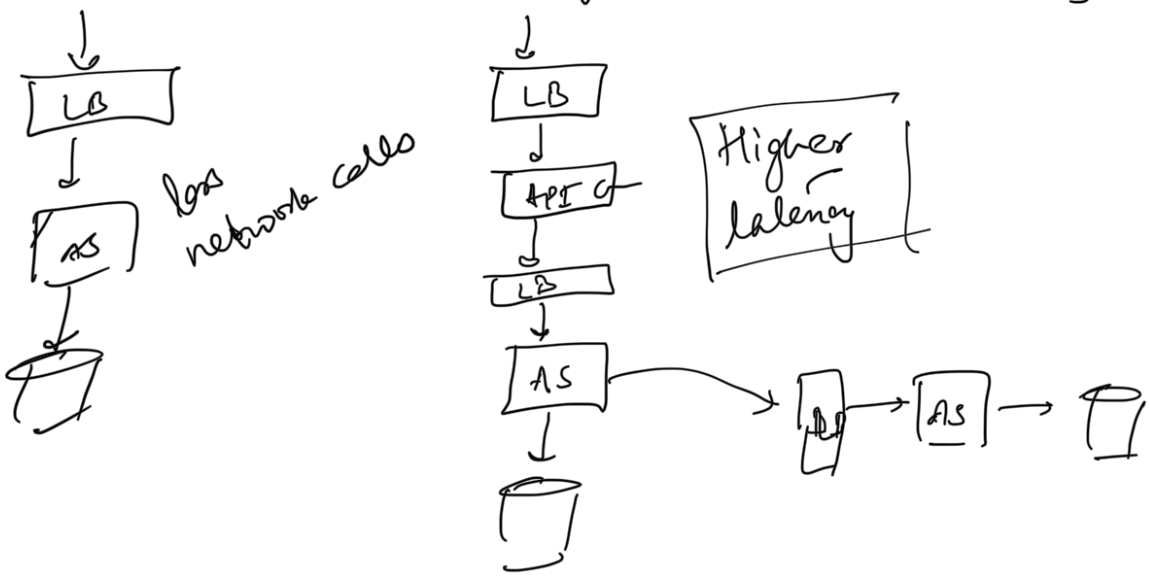

Facebook
News Feed
Ads
Photos
Text
Videos
Events
PYMK
{...} → API
newsfeed(userid, page_n)
News Feed
Ads
Events
PYMK
Micro1
/ads(userid, loc...)
Micro 2
/events(user_id, ...)
Micro3
/pymk(userid)


① Separation of concern
→ tech stack/DB
→ deploying code
→ testing
→ Scaling
→ robustness
User id
name
address
username
password
Products
100x
Users
1 million

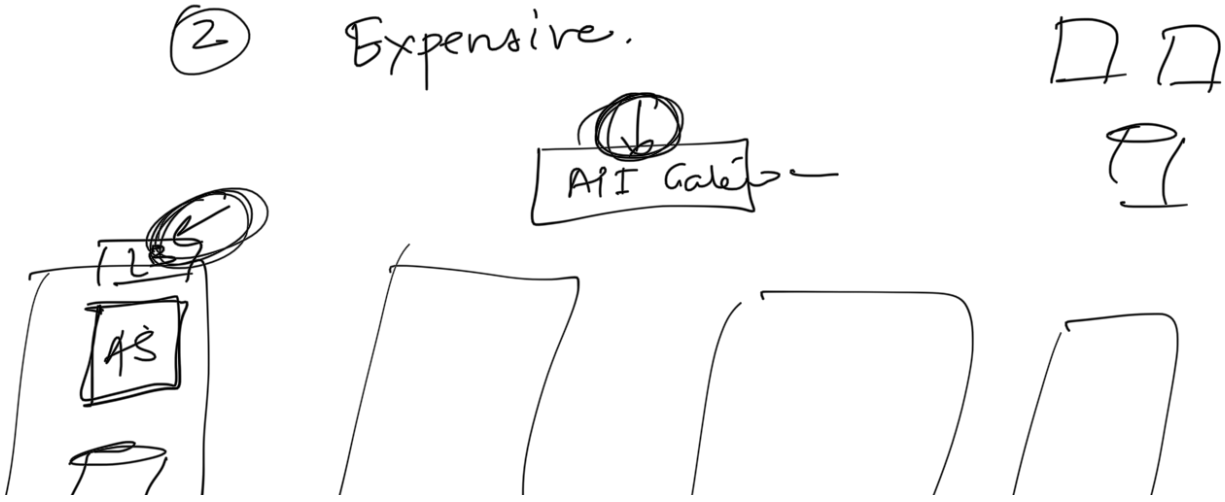


Cons:

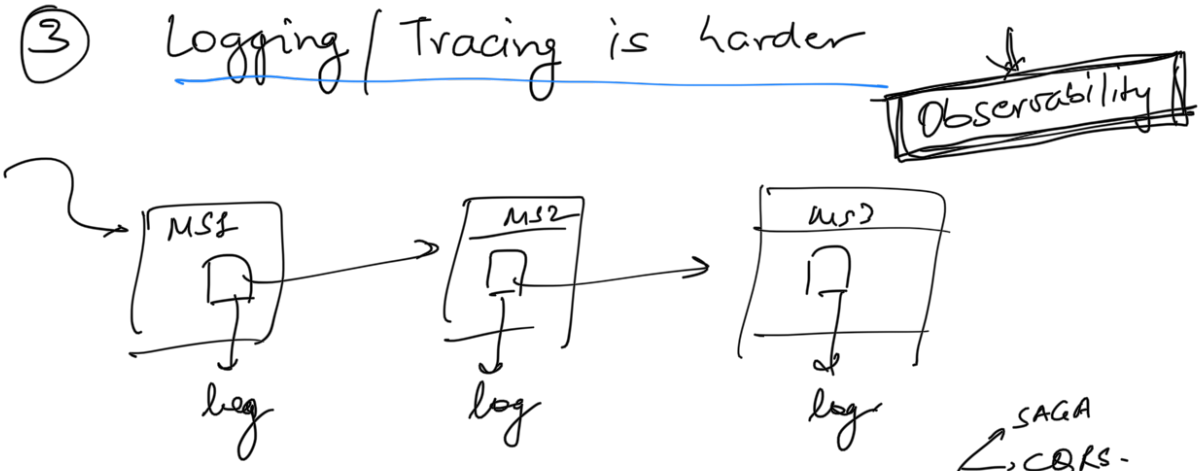
① more layers → more latency



② Expensive.

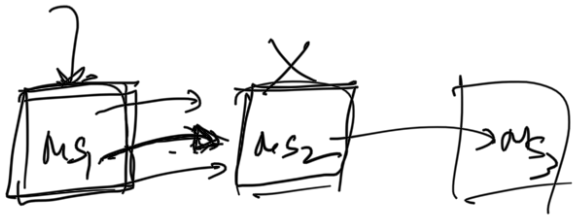


③ Logging / Tracing is harder



④ Data inconsistency → distributed transaction

SAGA, CQRS.



Circuit-breaker pattern

Class 2 ① How can I reduce latency

↳ event driven architec—

② Observability in Microservices

Next class ③ Distributed Transactions in Microse—

SAGA CQRS

Class 2 ④ Handle "Service Down" gracefully

↳ Circuit Breaker Pattern.

3 ⑦ Payment Service ⑤ Authentication in

Charts

② Notif S

API Gateways

class 3

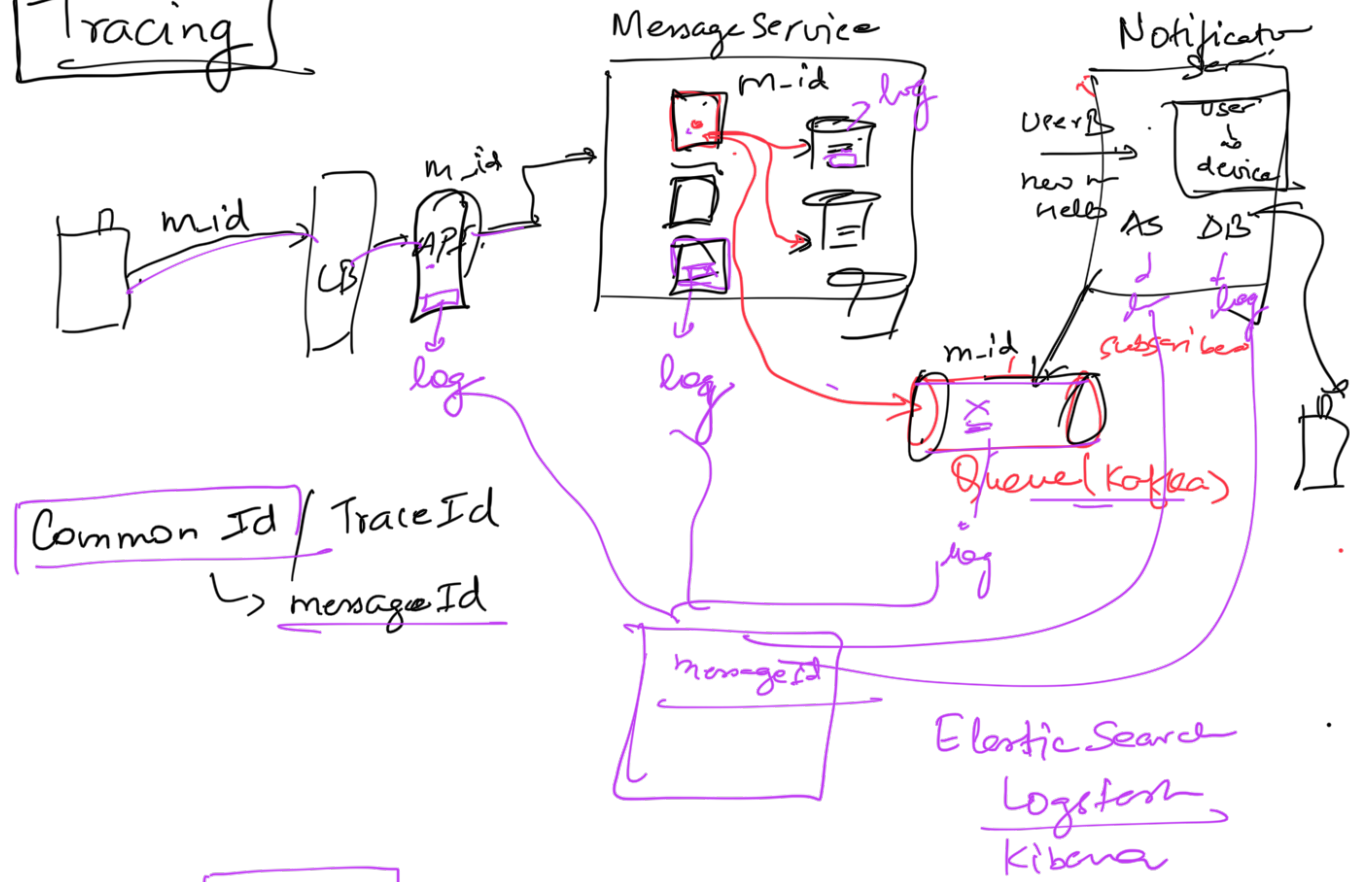
Observability

① Detection → Alarm
↳ Notification

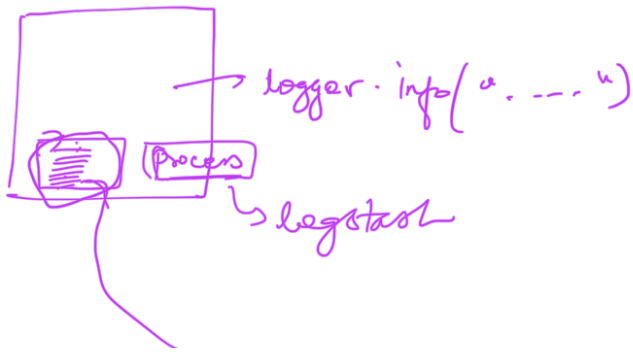
② Logs or metrics
↳ judge health
or diagnose
→ verify fix

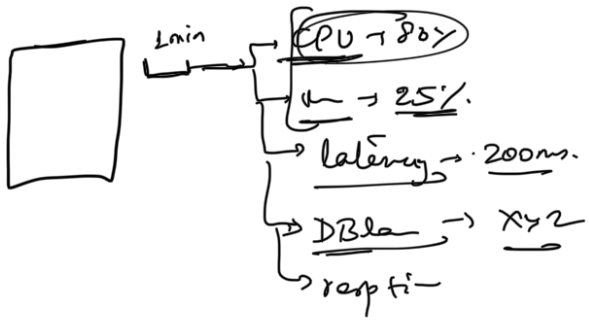
Distributed Tracing

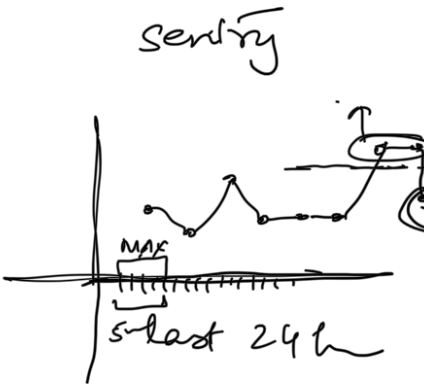
10 hours

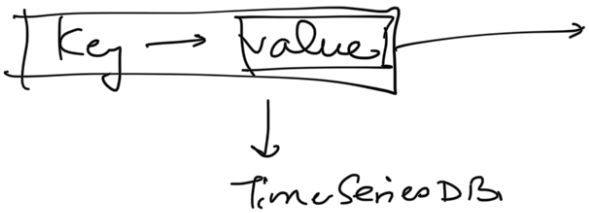


① Common Id / TraceId
↳ messageId



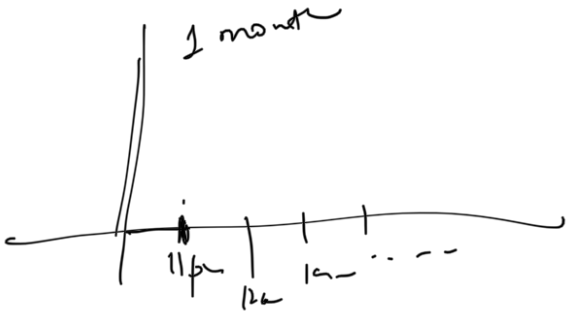






aggregated
- 1 min
- 5 —
- daily

Key	value	min
		1
		2
		3
		4
		5
		⋮
		86400

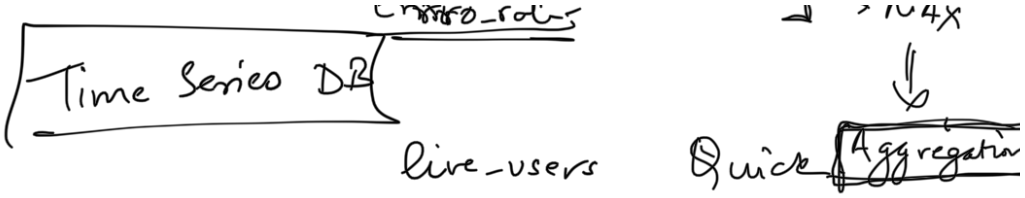


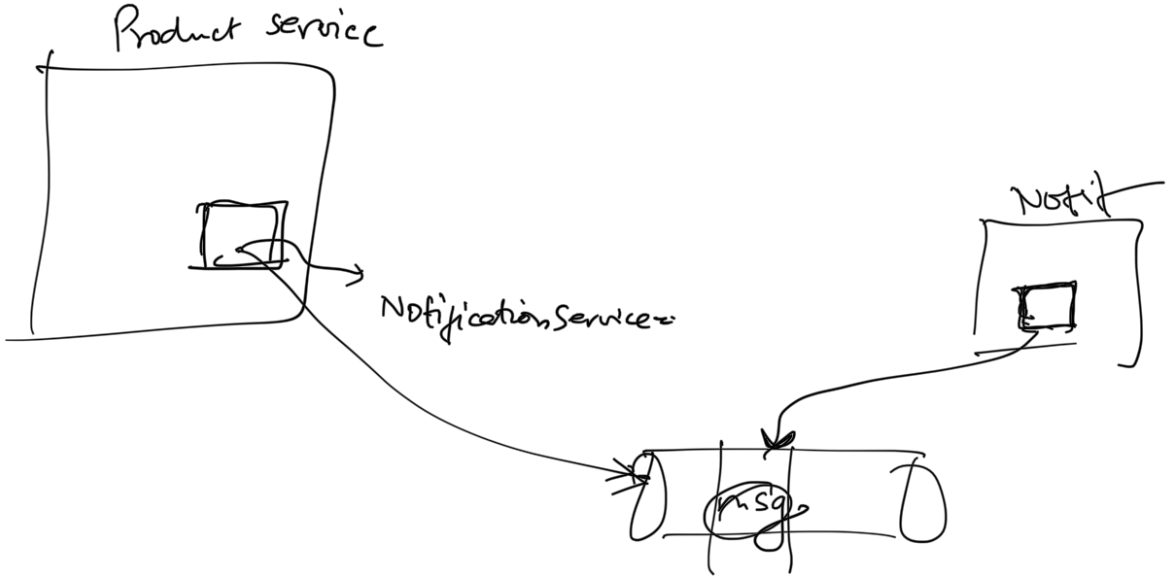
ods.publish("live_users", ~~100~~ value);

Latency

11:01	200
11:02	400
11:03	600
⋮	

→ SUM
→ AVG





struct { obj { "text" : "Hello"

"sender" : "- - - -"

"messageId" : - - - }